Table of Contents

Systems Reference

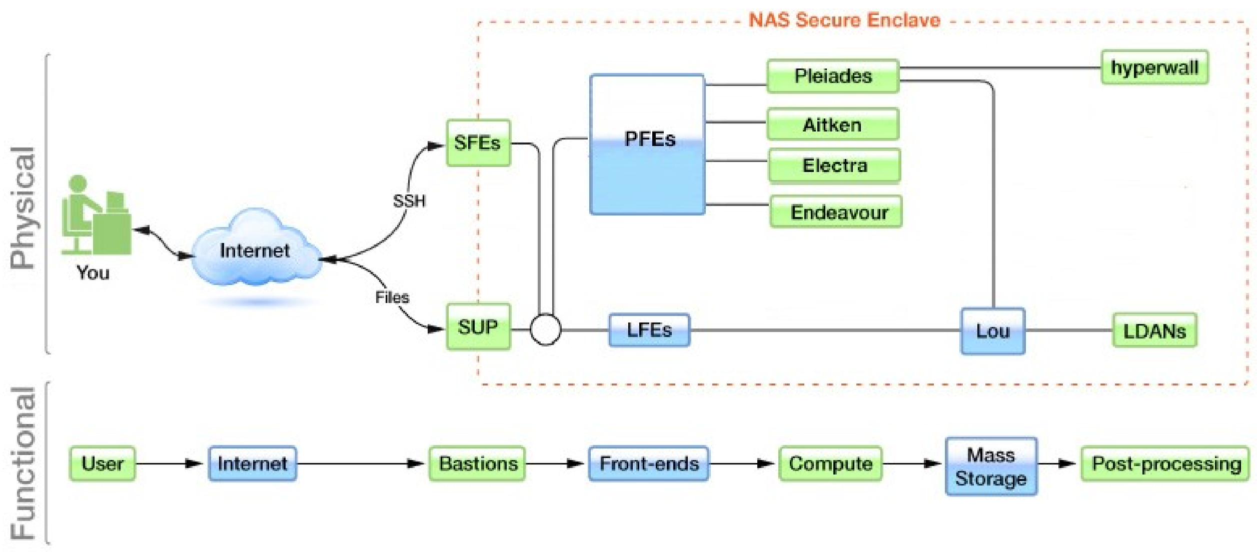
HPC Environment Overview

Our HPC environment is operated by staff in the NASA Advanced Supercomputing (NAS) Division at Ames Research Center located at Moffett Field, CA. The supercomputers and support staff are funded by NASA's High-End Computing Capability (HECC) Project.

The topics in this section summarize the system components, such as the compute nodes, secure network connections, front-end systems, and data storage facilities; and the user environment, including information about the Linux operating system and the various filesystem directories that are available for your use.

Systems Overview

The systems available for your use are all protected within a secure environment. The figure below shows both the physical connections among all the components and their functional relationships through a typical user workflow. The environment is described in more detail in the following sections.



The Secure Enclave

NAS supercomputing components are protected within a secure enclave that can be accessed only by authenticated users through the following secure bastions:

- Secure front ends (SFEs)
- Secure Unattended Proxy (SUP)

Components protected within the secure enclave include:

- Supercomputers: Pleiades, Aitken, Electra, Endeavour
- Front-end nodes: Pleiades, Aitken, Electra, Endeavour (PFEs), Lou (LFEs)
- Lou mass storage system

- Lou data analysis nodes (LDANs)
- hyperwall visualization system

The following sections give an overview of each component of the secure enclave.

Bastions

Secure Front Ends (SFEs)

The secure front ends (SFEs) provide inbound connection from your local system to the secure enclave. The first time you access the HECC systems within the enclave, you will authenticate through an SFE. Subsequently, you can use any of the bastions to access systems in the enclave.

For an overview of the initial authentication process, see Logging in for the First Time. For more information about the SFEs, see the article Role of the Secure Front Ends.

Secure Unattended Proxy (SUP)

The Secure Unattended Proxy (SUP) allows you to pre-authenticate to the secure enclave for one-week periods, during which you can perform unattended (batch) file transfers. After you complete the setup process, SUP is the most efficient and convenient method for transferring files from your remote system. For more information, see Using the Secure Unattended Proxy (SUP).

Front Ends

The HECC supercomputers Pleiades, Aitken, Electra, and Endeavour share the Pleiades front-end systems (PFEs). You can use the PFEs to edit files, compile your code, run short debugging and testing sessions, and submit batch jobs to the Pleiades, Aitken, or Electra compute nodes or to Endeavour. See the following articles for more information:

- Pleiades Front-End Usage Guidelines
- Pleiades Front-End Load Balancer

Compute Nodes

There are currently four supercomputers available for users: Pleiades, Aitken, Electra, and Endeavour.

Pleiades

NASA's flagship supercomputer, and one of the most powerful production systems in the world, Pleiades is an HPE/SGI ICE cluster containing multiple generations of Intel processors. See the following articles for more information:

- Pleiades Resource Page
- Pleiades configuration and usage guidelines

Aitken

Aitken, NASA's newest supercomputer, is housed in the Modular Supercomputing Facility, an environmentally-friendly module located a short distance from the main NAS building. Aitken uses the Pleiades front ends (PFEs), filesystems, PBS server, and job queues. See the following articles for more information:

- Aitken Resource Page
- Aitken Configuration Details
- Preparing to Run on Aitken Cascade Lake Nodes
- Preparing to Run on Aitken Rome Nodes

Electra

Electra is NASA's first prototype modular supercomputing system, housed near the main NAS building. Electra uses the Pleiades front ends (PFEs), filesystems, PBS server, and job queues. See the following articles for more information:

- Electra Resource Page
- Electra Configuration Details
- Preparing to Run on Electra Skylake Nodes
- Preparing to Run on Electra Broadwell Nodes

Endeavour

Endeavour is an HPE Superdome Flex system that provides resources for user applications needing access to large cache-coherent, global shared-memory capabilities in a single system image (SSI). Endeavour uses the Pleiades front ends (PFEs) and filesystems, and shares some of the Pleiades InfiniBand fabric. However, Endeavour uses its own designated Portable Batch System (PBS) server and job queues. See the following articles for more information:

- Endeavour Resource Page
- Endeavour Configuration Details
- Preparing to Run on Endeavour

Mass Storage System

The NAS facility provides long-term storage space on a single mass storage system, known as Lou. This HPE/SGI system has 7.6 petabytes (PB) of disk space and is capable of storing up to 1040 PB (1 exabyte) on tape. See the following articles for more information:

- Your Mass Storage Directory
- Lou Mass Storage System

Post-Processing Systems

Systems provided for post-processing include the Lou data analysis nodes (LDANs), designated as ldan[*11-14*], and the hyperwall visualization system. For a summary on using these systems for post-processing work, see Post-Processing Your Data. For detailed information, see also the following articles:

- Pleiades Front-End Usage Guidelines
- Lou Data Analysis Nodes
- Visualization System: hyperwall

Networks

The NAS high-speed network (NASLAN) includes a 10 gigabit-per-second (Gb/s) local area network and 10 Gb/s peering with other high-speed networks such as the NASA Integrated Communications Services (NICS), Internet2, and the Consortium for Educational Networks in California (CENIC). For an overview, see Networking Resources.

To access the HECC resources inside the secure enclave, you will use the SSH protocol to connect from your desktop system to a bastion (usually the SFEs) through a wide area network and the NASLAN.

User Environment

All HECC systems run the Linux operating system. If you are new to Linux, you can find a lot of helpful information at the user-supported community website Linux.org, including a Beginners Learning Course that provides instruction on the basic directory structure of Linux, how to get around in the directories, how to access Linux documentation (man pages), useful commands, and much more. You can also find support at the Linux forum.

When your NAS account is created, your default Linux shell is set to be the C shell (csh); this is assumed to be the case throughout this guide. If you want to use a different shell as your default, such as bash, call the NAS Control Room staff at (800) 331-8737 or (650) 604-4444 or send an email message to support@nas.nasa.gov to request the change. After the change is made, the new default shell of your choice applies to all of your jobs.

Once you complete the initial setup for your NAS account, you will have access to the Pleiades front-end (PFE) systems, the home filesystems, the Lou mass storage filesystems, and the scratch (/nobackup) filesystems. Your NAS account is authorized to run jobs on all HECC compute systems.

NAS supercomputers use the Portable Batch System (PBS) from Altair Engineering, Inc., for job submission, monitoring, and management. For more information about PBS, see Submitting and Running Jobs.

Filesystems

Pleiades, Aitken, Electra, and Endeavour share the same home and scratch (/nobackup) filesystems. When you log into a PFE, you will have access to the following directories:

- A home directory on the Pleiades home filesystem, which you can use to store a small number of files such as source code, small input files, and so on
- A /nobackup directory on a Lustre filesystem, which you can use to temporarily store larger files for reading and writing large amounts of data while running jobs

For long-term data storage, you also have access to a home directory on the Lou mass storage systems. The /nobackup filesystems are mounted on Lou, so you can easily copy files there directly from your /nobackup directory.

The Pleiades and Lou home filesystems are backed up each night. These backups are stored for approximately one year. The scratch (/nobackup) filesystems are *not* backed up.

Quota limits are enforced on all filesystems. Two kinds of quotas are supported:

- Limits on the total disk space occupied by your files
- Limits on the number of files you can store, irrespective of size; for quota purposes, directories count as files

See Quota Policy on Disk Space and Files for more information.

Your Home Directory

Your home directory is located on the Pleiades home filesystem, which is accessible from Pleiades, Aitken, Electra, and Endeavour. Use your home directory to store a limited number of smaller files such as source code and input files. For temporary, short-term storage of larger files, use your /nobackup directory. For long-term data storage, use the Lou mass storage systems. See Pleiades Home Filesystem for more information.

Your Scratch (/nobackup) Directory

Use your /nobackup directory to temporarily store large files that read and write large amounts of data when you run jobs. Your /nobackup directory resides on one of several Lustre filesystems, designated /nobackupp*X*. To find out which Lustre filesystem your /nobackup directory is located on, run:

```
pfe21% ls -ld /nobackup/your_nas_username
```

The /nobackup filesystems are also mounted on the Lou mass storage system, so you can access data in your /nobackup directory from Lou without going through Pleiades, Aitken, Electra, or Endeavour.

WARNING: The /nobackup filesystems mean just that: they are not backed up. While this is stating the obvious, some users have lost important data by storing it on these systems over long periods of time. It is your responsibility to copy essential data to either your home directory, to archival storage on the Lou systems, or to your remote system.
See Pleiades Lustre Filesystems for more information.

Your Mass Storage Directory

For safe, long-term data storage, transfer your files to your Lou home directory. The Lou mass storage filesystem allows you to retrieve your stored files quickly and securely whenever you need them.

You can log into the Lou system just as you would any other HECC system and save data to mass storage by copying your files to your Lou home directory: Lou:/u/*your_nas_username*.

There is no specified data-size quota for your Lou home directory, but you can store up to 250,000 files.

For more information, see The Lou Mass Storage System.

The Data Migration Facility

Data stored on Lou is migrated to tape, as needed, to make space on the disks for more data. Migrated files are retrieved to active disk when you attempt to read or write to them. These migration and retrieval processes are managed by HPE/SGI's Data Migration Facility (DMF), which also enables you to manually list, put, find, and get files that are on tape.

When your data is migrated to tape, two copies are written to two separate tape media in automated tape libraries located in two different buildings. See Data Migration Facility Commands for more information.

The Lou Filesystem

Lou is composed of the Lou front ends (LFEs), designated lfe[*5-8*]. The /nobackup filesystems are mounted on the LFEs, so you can easily copy files there directly from your /nobackup directory.

Although you cannot perform post-processing tasks on the LFEs, the Lou data analysis nodes provide PBS resources to perform post-processing tasks on your Lou mass storage data. For more information, see the following articles:

- The Lou Mass Storage System
- Lou Data Analysis Nodes

Software Modules and Packages

HECC provides many software programs such as compilers, pre- and post-processing programs, analysis tools, and math and scientific libraries. The software is managed through the use of packages and modules that you can load into your home directories.

For more information, see the following articles:

- Software on NAS Systems
- Customizing Your UNIX Environment

Front-End Systems

Pleiades Front-End (PFE) Usage Guidelines

The PFEs are the front-end systems for Pleiades, Aitken, Electra, and Endeavour. They provide an environment that enables quick turnaround for tasks such as file editing, file transferring, compiling, and short debugging/testing sessions, as well as for batch job submissions via PBS to a subset of the Pleiades compute nodes, or to Aitken, Electra, to Endeavour.

WARNING: The PFEs use Intel Sandy Bridge processors. If you use a PGI compiler to build your executable on the PFEs, be aware that by default the executable is optimized for the Sandy Bridge microarchitecture (which includes Sandy Bridge and Ivy Bridge) and will not necessarily execute on the compute node processors. See PGI Compilers and Tools for information on generating a single executable that will work on all processor types.
You cannot use SSH to access the compute nodes except for the subset of nodes your PBS job is running on.

Pre-Processing and Post-Processing Data

For pre- and post-processing applications such as Tecplot, IDL, and MATLAB, we recommend using the Lou data analysis nodes (LDANs). You can request the LDANs in the **qsub** command line. The Lou home filesystems, Pleiades home filesystems, and /nobackup filesystems are all mounted on the LDANs.

Restrictions on Front-End Systems

MPI (Message Passing Interface) jobs are *not* permitted to run on the PFEs. In addition, you will be notified by email if a job running on a PFE exceeds 27 GB.

Before starting a large-memory session, it is a good idea to make sure there is enough memory available. You can run the **top** command, hit "M", and check under the "RES" column for other large memory applications that may be running.

File Transfers to Mass Storage

The /nobackup filesystems are mounted on Lou, so the easiest way to transfer files between Pleiades and Lou is to initiate a command such as **shiftc**, **cp**, **mcp**, or **tar** on Lou. For example:

```
lou% shiftc /nobackup/username/filename $HOME
```

If you initiate the transfer from Pleiades, you can use the commands **scp** or **shiftc** to transfer files between a PFE and Lou. For very large file transfers, we recommend the NAS-developed Shift tool (**shiftc**).

File transfers from the compute nodes to Lou must first go through one of the PFEs.

When sending data to Lou, keep your largest individual file size under 1 terabyte (TB). Files larger than 1 TB will occupy all of the tape drives, preventing other file restores and backups.

Role of the Secure Front Ends

The secure front ends (SFEs) are the bastion hosts protecting the NAS secure computing enclave. The enclave includes all of the major HECC systems, including: Pleiades, Aitken, Electra, Endeavour, Lou, and the hyperwall.

To access resources inside the enclave from your local system, you must first use Secure Shell (SSH) to connect to one of the SFEs (sfe[*6-9*]) and log in using one of the following two-factor authentication methods:

- RSA SecurID + NAS password
- RSA SecurID + NASA personal identity verification (PIV)
- RSA SecurID + public key

Once authenticated, you can then use SSH to access any of the NAS systems from the SFE.

You can avoid having to log in twice (first to an SFE and then to a system inside the enclave) by setting up SSH Passthrough.

Note: Using SSH to connect from the SFEs to hosts outside of the enclave is not allowed.

2021 SFE Replacement: Actions and Changes

On March 19, 2021, four new SFEs, sfe[6-9], were deployed. The old systems, sfe[1-3], were decommissioned on Friday, April 5, 2021. If you have not yet transitioned to the new SFEs, please complete the steps below.

Transitioning from sfe[1-3] to sfe[6-9]

You must change your ~/.ssh/config configuration file on all hosts that you use to access the current SFEs. Old configurations are no longer valid due to changes in host names and changes to the SSH ciphers and algorithms supported. You should also transition any unrelated configuration settings that you may have in your existing ~/.ssh/config file.

You can download a new NAS configuration file template here: ssh_config.txt

Rename your old ~/.ssh/config file to a different name (such as config.pre_sfe6) before you download the new file, in case you need to transfer any settings from your old file to the new file.

Note: Public keys that were previously set up for SSH passthrough on the old SFEs have been copied over to the new SFEs, so you do not need to re-upload them.

New SFE Features and Changes

The new SFEs have some new features, as well as some changed functionality:

- The new SFEs are built on a different and more restrictive architecture than the old ones. You can no longer directly read or write any file on the SFEs. You can only execute the commands **`ssh`**, **`ssh-balance`**, and a new command called **`ssh-key-init`**.
- SSH public keys used for SSH passthrough are managed using the new **`ssh-key-init`** command. For instructions on using this command to initialize public keys, see: Setting

Up Public Key Authentication

- Transfers to and from the SFEs themselves are no longer allowed. Existing mechanisms using SSH passthrough and SUP/Shift are still supported as previously. For transfer instructions, see: Inbound File Transfer Through SFEs: Examples
- When you log into an SFE using the RSA SecurID passcode + NAS password method, the password will now be prompted *before* the RSA SecurID passcode (instead of after).
- A new authentication method, RSA SecurID passcode + NASA PIV, is now available for users who hold NASA personal identity verification (PIV) badges. When your NASA badge is attached to your workstation via a card reader, this method enables you to log in using your PIV personal identification number (PIN) and RSA SecurID passcode.

Lou Mass Storage

The Lou Mass Storage System

For safe, long-term data storage, transfer your files to the Lou mass storage system, which allows you to retrieve your stored files quickly and securely whenever you need them.

The Lou mass storage system is an Intel Xeon Gold 6154 "Skylake"-based cluster combined with a Spectra Logic tape storage archive. Lou uses a parallel Data Migration Facility (DMF) system to provide high speed and bandwidth for data transfers between Lou's disks and tapes.

Connecting to Lou

Lou's four hosts, known as Lou front-end systems (LFEs), are designated as lfe[*5-8*].

You can connect automatically to the LFE that has the lowest load by using the "lou" or "lfe" hostnames in your **`ssh`** command line: **`ssh lfe`** or **`ssh lou.`** This behavior is similar to the load balancing process that occurs on the Pleiades front-end nodes (PFEs) when you run **`ssh pfe`**.

Transferring Files Between /nobackup Filesystems and Lou

The Lustre (/nobackup) filesystems are mounted on the LFEs. To transfer files between your /nobackup filesystem and your Lou home filesystem, use the local file transfer commands **`shiftc, cp`**, or **`mcp`**. For example:

```
lfe% cp /nobackup/your_username/filename /u/your_username
```

You can also create a tar file that contains the data in one of your /nobackup subdirectories and then store the tar file in the LFE home filesystem. For example:

```
lfe% cd /nobackup/your_username
lfe% tar cf /u/your_username/mydir.tar mydir
```

Transferring Files from Your Local System to Lou

Recommended: Use the Shift tool (**`shiftc`**), which accepts both the alias and specific hostnames: lou, lfe, or lfe[*5-8*].

For more information about transferring files to and from your local system, see Remote File Transfer Commands.

Data Migration Between Disk and Tapes

In addition to 7.6 petabytes of disk space, Lou has 51 LTO-8 tape drives. Each LTO-8 tape holds 12 TB of uncompressed data, for a total storage capacity of approximately 1040 petabytes, or one exabyte (with normal 35% compression). Data migration (from disk to tape) and unmigration (from tape to disk) are managed by the Data Migration Facility (DMF).

Data stored on Lou's home filesystems (on disk) are automatically migrated to tape. Two copies of your data are written to tape media in silos located in separate buildings. When it is necessary to make room for more data, some files that have been written to tape may have their data "released" from disk. This means that the file is still visible on the filesystem, but the data must be retrieved from tape before the file can be used.

When a file completes writing to tape on Lou, its **`ctime`** attribute is updated to signal that it is available to be backed up by system software. This should not have an affect on any file transfer applications such as **`rsync`**.

If you need to retrieve data that is on tape, be sure to unmigrate the data from tape to your home filesystem on Lou before transferring it to other systems.

TIP: If you use the Shift tool (**`shiftc`**) for file transfers, it will automatically ensure that files on Lou are online before the transfer.
If you are not using Shift, you can use the following DMF commands to retrieve your files from tape:

```
$ dmls -al file1 file2 . . .    # show the status of your files.
$ dmget file1 file2 . . . &     # retrieve your file from tape.
```

At this point, you can start your transfer and the files will transfer as they come online.

WARNING: Do not use the /nobackup filesystems for long-term data storage. As the names suggest, these filesystems are not backed up, so any files that are removed cannot be restored. You should store essential data on more permanent storage devices, such as Lou.
For more tips on how to use the Lou storage systems more effectively, see:

- Portable File Names and Sizes
- Dealing with Slow File Retrieval

Post-Processing Your Data

To perform post-processing tasks on Lou data, use the Lou data analysis nodes, which provide dedicated PBS resources for that purpose.

Post-processing is not permitted on the LFEs. To enforce this, a system monitoring tool called **`query_wms`** runs on the LFEs and kills any user process that uses more than 1 GB of memory.

Quota Limits On Lou

There are no disk quota limits on your Lou home filesystem. However, there *are* limits on the number of files (inodes):

- 250,000 inode soft limit (14-day grace period)
- 300,000 inode hard limit

To check your quota status and usage on your home filesystem, log into Lou (lfe) and run the **`quota -ls`** command as follows:

```
lfe% quota -ls
Disk quotas for user username (uid xxxx):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
/dev/cxvm/sfa2_s2n
                 87422M       0       0           59381    250k    300k
```

See Quota Policy on Disk Space and Files for more information about HECC system quotas.

Lou Data Analysis Nodes

The Lou data analysis nodes (LDANs) provide dedicated PBS resources to perform post-processing tasks on your Lou mass storage data. You can also use them for performing pre-processing or post-processing tasks on active Pleiades, Aitken, or Electra data. The LDANs can be accessed only through PBS jobs in which resources are dedicated to the job owner.

The LDANs contain Intel Xeon Gold 6154 "Skylake" processors, equipped with more memory than the Electra Skylake nodes, as follows:

- Each ldan[*11-12*] has 36 cores and 768 GB of memory
- Each ldan[*13-14*] has 36 cores and 1.5 TB of memory

The Lou, Lustre, and Pleiades home filesystems are mounted on the LDANs. You can access them in a PBS job running on an LDAN by using the following paths:

Lou:
    /u/*username*
Lustre:
    /nobackup/*username*
Pleiades:
    /pleiades/u/*username*

Software Modules

All of the software packages that you use on the Pleiades front-end (PFE) nodes or Pleiades/Aitken/Electra compute nodes are also available on the LDANs. However, both the Lou front-end (LFE) nodes and LDANs use startup files from your Lou home directory rather than your Pleiades home directory.

Therefore, if your PBS jobs rely on loading specific software modules or environment variables in your Pleiades system startup files (such as .cshrc), in order to run them on the LDANs you must modify your startup files on Lou to load the modules and set appropriate environment variables.

Running PBS Jobs on LDANs

The PBS server for the LDANs is pbspl1. To use the LDANs, submit your jobs to the **`ldan`** queue. Each job can use only *one* LDAN for up to *three days*, and each user can have a maximum of *two jobs* running simultaneously.

Before You Begin: If you want to process archive data that has been migrated offline to tape, use the Data Migration Facility command **`dmget`** to migrate the data back to disk before submitting jobs to the LDANs.

Submitting Your PBS Job

You can submit interactive PBS jobs to the LDANs from either the LFEs or the PFEs. You can submit PBS job scripts from either your Lustre home filesystem (/nobackup/*username)* or your Lou home filesystem (/u/*username)*.

WARNING: Do not submit job scripts from the Pleiades home filesystem, because PBS error and output files cannot be copied back there from the LDANs.

Use the **:mem=*xxx*GB** attribute to specify how much memory you need for your job. This will instruct PBS to allocate an appropriate LDAN.

Note: Keep in mind that the amount of memory on a node that is available for your job is slightly less than the total physical memory, because the system kernel can use up to 4 GB of memory in each node.

Sample PBS Script

```
#PBS -lselect=1:ncpus=36:mem=750GB
#PBS -lwalltime=2:00:00
#PBS -q ldan

### Remember to load necessary modules
### or set environment variables that you need

module load xxx

### Note that the default directory a PBS job
### is in when it starts is the Lou home filesystem.
### Change directory to the appropriate directory
### for your pre- or post- processing work.

cd $PBS_O_WORKDIR

### If your executable for pre- or post- processing
### is located under your Pleiades home filesystem,
### use the pathname /pleiades/u/username/bin/....
### do not use the pathname /u/username/bin/...

/pleiades/u/username/bin/your_processing_code < input > output
```

Running Graphics Applications on LDANs

If you want to run graphics applications that consume enough memory to require one of the larger-memory LDANs, submit an interactive PBS job from one of the LFEs that specifies **mem=1040GB**. For example:

```
lfe% qsub -I -q ldan -lselect=1:mem=1040GB,walltime=1:00:00
```

This will provide you with an LDAN for the duration of the specified wall time. Then, from any other terminal window, you can use SSH to connect directly to the LDAN assigned to your job, and run your graphics application interactively. If you have SSH passthrough set up, you can even SSH directly in to the LDAN from your workstation by adding the LDANs as allowed SSH passthrough targets in your .ssh/config file.

To run graphics applications directly from a PBS session submitted from an LFE, you need to export your DISPLAY environment by adding the **-X** option to the **qsub** command, as follows:

```
lfe% qsub -I -X -q ldan -lselect=1:mem=750GB,walltime=1:00:00
```

If you are a remote user, you can also explore using Virtual Network Computing (VNC) to connect to NAS systems.

Aitken

Aitken Configuration Details

Aitken (pronounced AY-ken) is a modular computing system housed in NASA's Modular Supercomputing Facility, located less than a mile from the main NAS building. Aitken is configured as follows:

Intel Cascade Lake-Based Compute Resources

- 4 E-Cells (8 E-racks)
- 1,152 Cascade Lake nodes
- 46,080 cores
- 221 terabytes (TB) total memory

AMD Rome-Based Compute Resources

- 16 racks
- 2,048 Rome nodes
- 262,144 cores
- 1.048 petabyte (PB) total memory

Aitken's total theoretical peak performance: 12.49 petaflops (PF)

Hostnames

For the Cascade Lake-based resources, there are four enclosures (individual rack units, or IRUs) in each E-rack, with 36 compute nodes per enclosure. For every two racks, there is a rack leader controlling them. The naming convention for the 288 nodes residing in every two racks (2 racks x 4 enclosures x 36 nodes) uses only odd rack numbers. Therefore, the hostnames of the Aitken compute nodes are r[x]i[0-7]n[0-35], where x are odd numbers between 901 and 907.

For the Rome-based resources, there are four enclosures in each rack, eight compute trays in each enclosure, and four nodes in each compute tray. The hostnames of the Aitken Rome nodes are r[x]c[1-4]t[1-8]n[1-4], where x are numbers between 201 - 216.

Processor, Memory, and Network Subsystems Statistics

The following table provides detailed configuration statistics for the processor, memory, and network subsystems for the Aitken compute nodes:

Aitken Processor, Memory, and Network Subsystems Statistics

Architecture	HPE SGI 8600-XA730i Gen10	HPE Apollo 9000
	Processor	
	Cascade Lake	**Rome**
CPU	20-Core Xeon Gold 6248	64-Core EPYC 7742
Newest Instruction Set	AVX-512	AVX2

Hyper-Threading	ON	OFF
TurboBoost	ON	ON
non-Turbo base CPU-Clock	2.5 GHz	2.25 GHz
Maximum Double Precision Floating Point Operations per Cycle per Core	32	16
# of Cores/node	40	128
Total # of Nodes	1,152	2,048
Total # of Cores	46,080	262,144
Total Double Precision TFlops	3,686	9,437
Memory		
L1 Cache	Local to each core; Instruction cache: 32K Data cache: 32K; Associativity: 8; Cache line size: 64B	Local to each core; Instruction cache: 32K Data cache: 32K; Associativity: 8; Cache line size: 64B

L2 Cache	1 MB per core;
	Associativity: 16;
	Cache line size: 64B
		512 KB per core;
		Associativity: 8;
		Cache line size: 64B

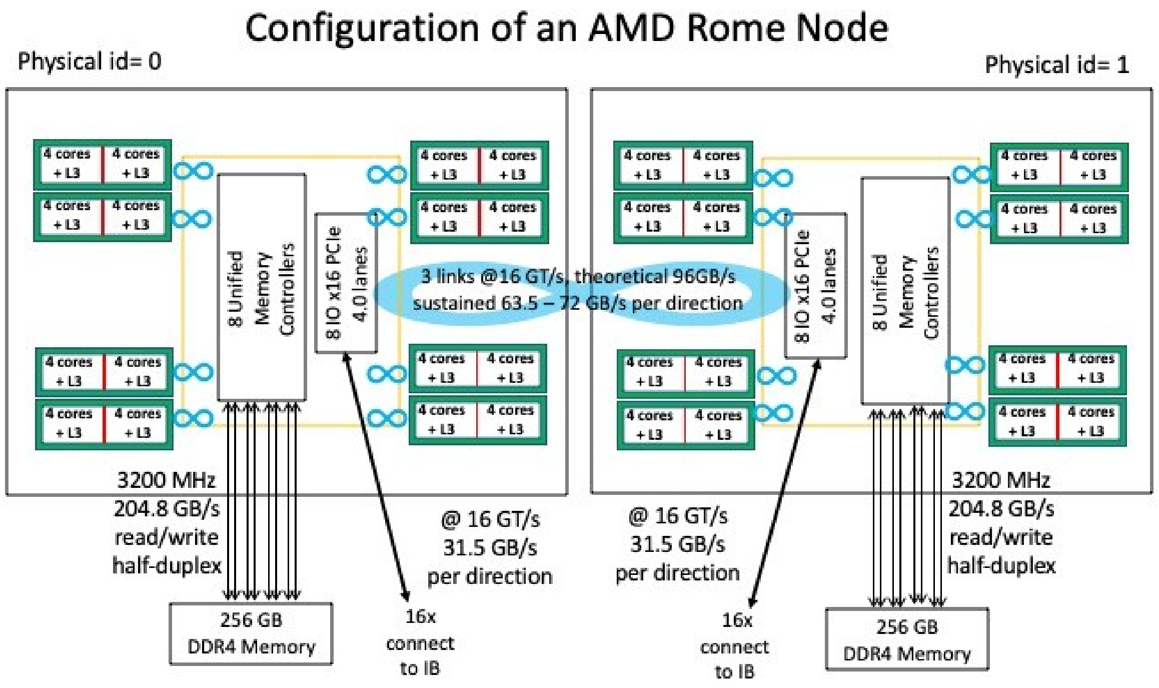
L3 Cache	27.5 MB shared non-inclusive by the 20 cores;
	Associativity: full;
	Cache line size: 64B
		16 MB shared among 4 cores in a core complex;
		256 MB per socket;
		Associativity: 16;
		Cache line size: 64B

TLB	Local to each core	Local to each core
Default Page Size	4 KB	4 KB
Memory/Core	4.8 GB; DDR4	4.0 GB; DDR4
Total Memory/node	192 GB	512 GB

Memory Speed and Bandwidth	2,933 MHz;
	6 channels;
	141 GB/sec read/write
		3,200 MHz;
		8 channels;
		204.8 GB/sec read/write

Inter-socket Interconnect	Ultra-Path Interconnect; 5.2 GHz, 10.4 GT/s, or 62.4 GB/sec	Global Memory Interconnect; 8.0 GHz, 16.0 GT/s, or 96.0 GB/sec
Inter-node Network		
IB Device on Node	Dual single-port 4x EDR IB Mezzanine card (2 single-port HCAs); 100 Gbits/s	Dual single-port 4x HDR IB Mezzanine card (2 single-port HCAs); 200 Gbits/s
IB Switches Between Nodes	4x HDR; 200 Gbits/s	4x HDR; 200 Gbits/s

AMD Rome Processors


Configuration of an AMD Rome Node
Physical id= 0
Physical id= 1
4 cores + L3
8 Unified Memory Controllers
8 IO x16 PCIe 4.0 lanes
3 links @16 GT/s, theoretical 96GB/s sustained 63.5 – 72 GB/s per direction
3200 MHz 204.8 GB/s read/write half-duplex
@ 16 GT/s 31.5 GB/s per direction
256 GB DDR4 Memory
16x connect to IB

This is a simplified configuration of an EPYC Rome node with two sockets. Each socket contains eight Core Complex Dies (CCDs, each enclosed in a green box) and one I/O die (IOD, enclosed in a yellow box). The infinity sign (â ¾) represents the Infinity Fabric. Each CCD contains two Core Complexes (CCXs). Each CCX has 4 cores and 16 MB of L3 cache. Thus, there are 64 cores per socket and 128 cores per node.

The AMD EPYC (pronounced "epic") 7742 Rome processor, incorporated into Aitken with the HPE Apollo 9000 system architecture, is in AMD's second generation System-on-Chip (SoC) processor family. Rome has the following high-level features.

- Zen 2 microarchitecture:

  The EPYC 7742 Rome processor has a base CPU clock of 2.25 GHz and a maximum boost clock of 3.4 GHz. There are eight processor dies (CCDs) with a total of 64 cores per socket.
- Hybrid multi-die design:

  Within each socket, the eight processor dies are fabricated on a 7 nanometer (nm) process, while the I/O die is fabricated on a 14 nm process. This design decision was made because the processor dies need the leading edge (and more expensive) 7 nm technology in order to reduce the amount of power and space needed to double the number of cores, and to add more cache, compared to the first-generation EPYC processors. The I/O die retains the less expensive, older 14 nm technology.
- 2nd-generation Infinity Fabric technology:

  Infinity Fabric technology is used for communication among different components throughout the node: within cores, between cores, between CCXs in a CCD, among CCDs in a socket, to the main memory and PCIe, and between the two sockets. The Rome processors are the first x86 systems to support 4th-generation PCIe, which delivers twice the I/O performance (to InfiniBand, storage, NVMe SSD, etc.) over 3rd-generation PCIe.

Processor Hierarchy

The Rome processor hierarchy is as follows:

- Core: A CPU core has private L1I, L1D, and L2 caches, which are shared by two hyperthreads on the core.
- CCX: A core complex includes four cores and a common L3 cache of 16 MB. Different CCXs do not share L3.
- CCD: A core complex die includes two CCXs and an Infinity Link to the I/O die (IOD). The CCDs connect to memory, I/O, and each other through the IOD.
- Socket: A socket includes eight CCDs (total of 64 cores), a common centralized I/O die (includes eight unified memory controllers and eight IO x16 PCIe 4.0 lanesâ total of 128 lanes), and a link to the network interface controller (NIC).
- Node: A node includes two sockets and a network interface controller (NIC). In an Aitken Rome node, the NIC is a dual single-port InfiniBand (IB) High Data Rate (HDR) 200 Gbit/s mezzanine card.

CPU Core

Rome is a 64-bit x86 server microprocessor. A partial list of instructions and features supported in Rome includes **`SSE, SSE2, SSE3, SSSE3, SSE4a, SSE4.1, SSE4.2, AES, FMA, AVX, AVX2`** (256 bit), Integrated x87 FPU (**`FPU`**), Multi-Precision Add-Carry (**`ADX`**), 16-bit Floating Point Conversion (**`F16C`**), and No-eXecute (**`NX`**). For a complete list, run **`cat /proc/cpuinfo`** on a Rome node.

Unlike the Intel Skylake and Cascade Lake processors, Rome does not come with the **`AVX-512`** instructions.

Each Rome core:

- Can sustain execution of four x86 instructions per cycle, using features such as the micro-op cache, advanced branch prediction, and prefetching. The prefetcher works on streaming data and on variable strides, allowing it to accelerate many different data structures.
- Has two 256-bit Fused Multiply-Add (FMA) units and can deliver up to 16 double-precision floating point operations (flops) per cycle. Thus, the peak double-precision flops per node is: 128 cores x 2.25 GHz x 16 = 4.6 teraflops. With 2,048 Rome nodes in Aitken, the peak is 9.44 petaflops.
- Can support Simultaneous Multi-threading (SMT), allowing two threads to execute simultaneously per core. SMT can be enabled/disabled in the Basic Input/Output System (BIOS) settings.

  Note: The final setting is yet to be decided. If you plan to take advantage of SMT, use **`cat proc/cpuinfo`** to determine whether it has been enabled.

Note: Linux support for the Zen microarchitecture started with Linux kernel 4.10. The SUSE Linux Enterprise Server Version 15 (SLES 15) operating system, which was previously used on the Aitken Rome nodes, uses Linux kernel 5.3. Although the current Red Hat-based operating system, Tri-Lab Operating System Stack Version 3 (TOSS 3), uses Linux kernel 3.10, it includes the needed patchesâ backported from the newer kernelsâ that are required to support the Rome nodes.

The AMD Optimizing C/C++ (AOCC), GNU, Intel, and PGI compilers can be used to compile codes for running on the Rome nodes. To generate optimized x64 code for the Zen 2 microarchitecture, consider using these compiler flags:

- AOCC compiler: `-march=znver2`
- Intel compiler: `-march=core-avx2` (preferred) or `-axCORE-AVX2`
- PGI compiler: `-tp=zen`
- GNU compiler (GCC): `-march=znver2`

  Note: `-march=znver2` is available in the `gcc 9.x` compiler and later versions. With `gcc 8.x`, use
  `-march=znver1` instead.

For more information, see Compiler Options Quick Reference Guide for AMD EPYC 7xx2 Series Processors.

Cache Hierarchy

The Rome cache hierarchy is as follows:

- op cache (OC): 4K ops, private to each core; 64 sets; 64 bytes/line; 8-way. OC holds instructions that have already been decoded into micro-operations (micro-ops). This is useful when the CPU repeatedly executes a loop of code. Using OC improves:
    - ♦ Pipeline latency: because the op cache pipeline is shorter than the traditional fetch and decode pipeline.
    - ♦ Bandwidth: because the maximum throughput from the op cache is eight instructions per cycle, whereas the maximum throughput from the traditional fetch and decode pipeline is four instructions per cycle.
    - ♦ Power: because there is no need to re-decode instructions.
- L1 instruction cache: 32 KB, private to each core; 64 bytes/line; 8-way. The processor fetches instructions from the instruction cache in 32-byte naturally aligned blocks.
- L1 data cache: 32 KB, private to each core; 64 bytes/line; 8-way; latency: 7-8 cycles for floating point and 4-5 cycles for integer; 2 x 256 bits/cycle load bandwidth to registers; 1 x 256 bits/cycle store bandwidth from registers; write-back policy.

  Note: With the write-back policy, data is updated in the current level cache first. The update in the next level storage is done later when the cache line is ready to be replaced.
- L2 cache: 512 KB, private to each core; unified; inclusive of L1 cache; 64 bytes/line; 8-way; latency: >= 12 cycles; 1 x 256 bits/cycle load bandwidth to L1 cache; 1 x 256 bits/cycle store bandwidth from L1 cache; write-back policy.
- L3 cache: 16 MB shared among four cores in a core complex (CCX); different CCXs do not share L3; total of 256 MB per socket. Within each CCX: 64 bytes/line; 16-way; latency: 39 cycles on average.

  Note: If a core misses in its local L2 and also in the L3, the shadow tags are consulted. If the shadow tag indicated that the data resides in another L2 within the CCX, a cache-to-cache transfer is initiated.
  1 x 256 bits/cycle load bandwidth to L2 of each core; 1 x 256 bits/cycle store bandwidth from L2 of each core; write-back policy; populated by L2 victims.

Memory Subsystem

Unlike the Zen 1 microarchitecture, in which each CCD has its own dual-channel memory controller, the Zen 2 microarchitecture places eight unified memory controllers in the centralized I/O die so that the memory latency is reduced (roughly at 200+ ns) and is more consistent. The memory channels can be split into one, two, or four Non-Uniform Memory Access (NUMA) Nodes per Socket (NPS1, NPS2, and NPS4). The BIOS default for HPE Apollo 9000 systems is NPS4, which is the highest memory bandwidth configuration geared toward HPC applications.

With eight 3,200-GHz memory channels, an 8-byte read or write operation taking place per cycle per channel results in a maximum total memory bandwidth of 204.8 GB/s per socket.

Each memory channel can be connected with up to two Double Data Rate (DDR) fourth-generation Dual In-line Memory Modules (DIMMs). For the Aitken Rome configuration, each channel is connected to a single 32-GB DDR4 registered DIMM (RDIMM) with error correcting code (ECC) support. In total, the amount of memory is 256 GB per socket and 512 GB per node.

Note: Using the one DIMM per Channel (DPC) configuration enables the system to run the memory DIMMs at the highest possible speed; a two-DPC configuration typically requires slightly reduced memory speed.

The memory frequency can be uncoupled, or it can be coupled with the Infinity Fabric frequency through BIOS settings to benefit either bandwidth-bound or latency-bound workloads. The uncoupled mode is used on the Aitken Rome nodes.

- Uncoupled Mode: For throughput-sensitive applications, to obtain higher read/write throughput, the Maximum Memory Bus Frequency option can be set to the maximum allowed (3,200 MT/s). In this case, the Memory Bus will not be synchronized optimally with the slower Infinity Fabric Clock, causing a slight increase in memory access latency.
- Coupled Mode: For latency sensitive applications, memory access latency can be reduced by setting the Maximum Memory Bus Frequency to 2933 MT/s or 2667 MT/s, in order to synchronize with the Infinity Fabric clock.

Intra-Socket Interconnect

The Infinity Fabric, evolved from AMD's previous generation HyperTransport interconnect, is a software-defined, scalable, coherent, and high-performance fabric. It uses sensors embedded in each die to scale control (Scalable Control Fabric, or SCF) and data flow (Scalable Data Fabric, or SDF).

- The SCF uses sensors to monitor die temperature, speed, and voltage across all cores within the dies and controls power management, security, reset, etc.
- The SDF connects the L3 caches to memory and to the configurable I/O lanes. SDF uses the configurable I/O lanes for memory-coherent communications between compute elements on a single die, between different dies on a socket, and between sockets in a node.
- The die-to-die Infinity Fabric bandwidth is 32 bytes for read and 16 bytes for write per Infinity Fabric clock (which has a maximum speed of 1,467 MHz).

Inter-Socket Interconnect

Two EPYC 7742 SoCs are interconnected via Socket to Socket Global Memory Interconnect (xGMI) links, part of the Infinity Fabric that connects all the components of the SoC together. In each Rome node configured with the HPE Apollo 9000 system architecture, there are 3 xGMI links using a total of 48 PCIe lanes. With the xGMI link speed set at 16 GT/s, the theoretical throughput for each direction is 96 GB/s (3 links x 16 GT/s x 2 bytes/transfer) without factoring in the encoding for xGMI, since there is no publication from AMD available. However, the expected efficiencies are 66-75%, so the sustained bandwidth per direction will be 63.5-72 GB/s.

Note: The xGMI link speed and width can be adjusted via BIOS setting. The xGMI Link Max Speed can be set to 10.667, 13, 16 or 18 GT/s. Setting it to a lower speed can save uncore power that can be used to increase core frequency or reduce overall power. It will also decrease cross-socket bandwidth and increase cross-socket latency. xGMI Dynamic Link Width Management saves power during periods of low socket-to-socket data traffic by reducing the number of active xGMI lanes per link from 16 to 8.

Inter-Node Network

There are 2,048 Rome nodes in the Aitken cluster. They are partitioned as follows:

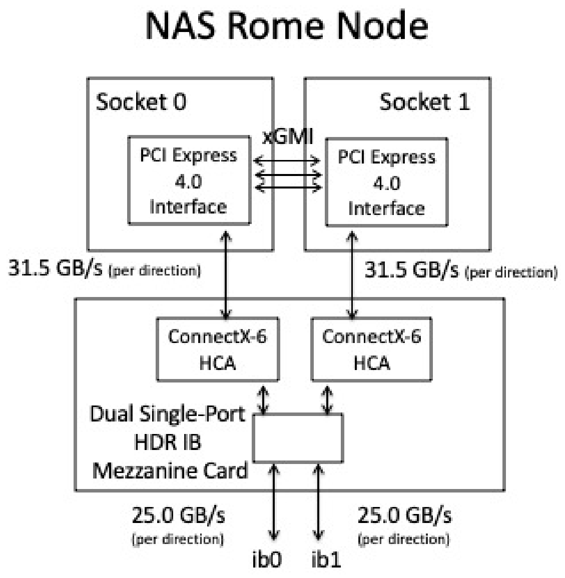
- 16 racks in the cluster (2,048 nodes total)
- 4 enclosures per rack (128 nodes/rack)
- 8 trays (HPE XL925g quad-node tray) per enclosure (32 nodes/enclosure)
- 4 nodes per compute tray

The hostname of each node is r[x1]c[x2]t[x3]n[x4], where x1 = 201 to 216; x2= 1 to 4; x3=1 to 8; and x4 =1 to 4.

The inter-node connection of Aitken's Rome nodes relies on the use of PCIe 4.0 lanes and one dual single-port IB HDR 200 Gbit/s mezzanine card on each node, and two IB HDR 200 Gbit/s premium switches per enclosure. One switch per enclosure facilitates the ib0 fabric, while the other facilitates the ib1 fabric.

As shown in the top diagram of this page, of the 128 PCIe 4.0 lanes per socket, 16 lanes are used to connect a node outwards. With a transfer rate of 16 GT/s, this enables a maximum bandwidth of 31.5 GB/s (when factoring in the 128/130 encoding) for each direction.

The two PCIe 4.0 x16 slots (one from each socket) are connected to the dual single-port IB HDR 200 Gbit/s mezzanine card, as shown below. The mezzanine card contains two separate Mellanox ConnectX-6 Host Channel Adapters (HCA), one for ib0 and the other for ib1, with a bandwidth of 25.0 GB/s per direction and a sub-microsecond latency. Note that the Mellanox ConnectX-6 200-Gbp/s adapters can achieve their maximum bandwidth (25.0 GB/s) in a PCIe 4.0 x16 slot (31.5 GB/s) but not in a PCIe 3.0 x16 slot (15.75 GB/s, as used in NAS Intel Xeon processors).


NAS Rome Node
Socket 0
Socket 1
xGMI
PCI Express 4.0 Interface
PCI Express 4.0 Interface
31.5 GB/s (per direction)
31.5 GB/s (per direction)
ConnectX-6 HCA
ConnectX-6 HCA
Dual Single-Port HDR IB Mezzanine Card
25.0 GB/s (per direction)
25.0 GB/s (per direction)
ib0
ib1

The 32 nodes in each enclosure are connected to one HPE Apollo 9000 IB HDR 40 Up premium switch (200 Gbit/s) for ib0, and to another for ib1. In each switch, there are two 40-port ConnectX-6 Application-Specific Integrated Circuit (ASIC) chipsâ 80 ports total, of which 32 are used as downlinks to connect to the 32 nodes in the enclosure, 40 are used as uplinks to connect to external targets, and four are used internally to connect between the two ASICs.

For each IB fabricâ with 32 nodes in an enclosure connecting to the same IB switch, forming the first-dimension in the topologyâ the 2,048 Rome nodes form a 7-dimensional enhanced hypercube.

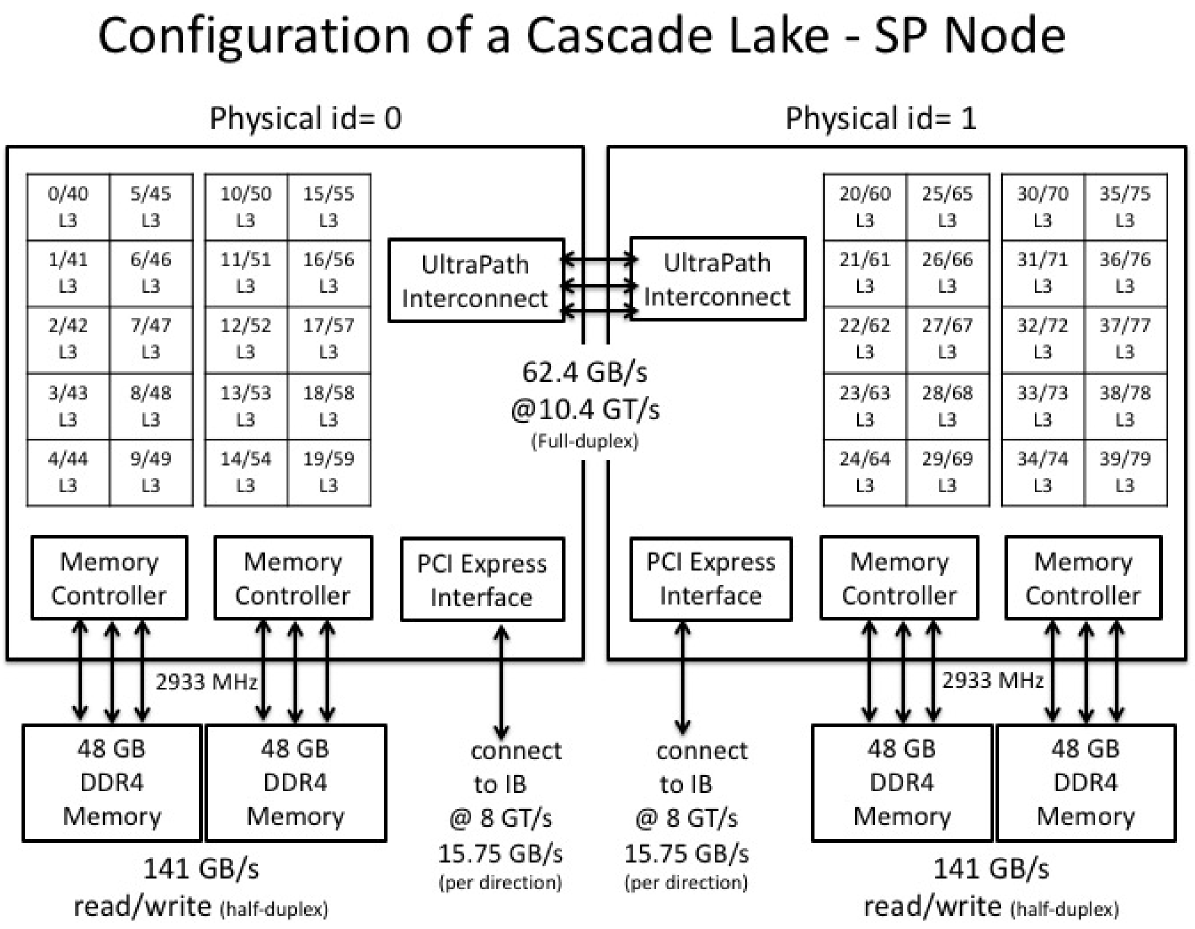
Connection Between Rome and Cascade Lake Nodes

The Rome and Cascade Lake nodes in Aitken will be connected with additional HDR cables on the 8th dimension.

References

- AMD EPYC 7742
- AMD EPYC 7742 specifications
- Compiler Options Quick Ref Guide for AMD EPYC 7xx2 Series Processors (PDF)
- High Performance Computing: Tuning Guide for AMD EPYC 7002 Series Processors (PDF)
- Workload Tuning Guide for AMD EPYC 7002 Series Processor Based Servers (PDF)
- Software Optimization Guide for AMD Family 17h Models 30h and Greater Processors
- AMD Infinity Architecture

Cascade Lake Processors



Notes:

- Each small square in the diagram represents a combination of a physical core and a L3 cache slice. For each physical core, there are two logical core labelings obtained from the **"processor"** entry of the `cat /proc/cpuinfo` output.
- Although Cascade Lake processors support up to three Intel Ultra Path Interconnect (UPI) links with a bandwidth of 62.4 GB/s, the HPE 8600 Saxon board used for the Cascade Lake nodes at NAS implements only two links, with a bandwidth of 41.6 GB/s.

The Intel Cascade Lake processor incorporated into the Aitken cluster is the 20-core Xeon Gold 6248 model. Its base clock speed is 2.5 GHz for non-AVX, 1.9 GHz for AVX2, and 1.6 GHz for AVX-512. It uses an enhanced 14-nanometer (nm) fabrication process and the Skylake microarchitecture with some optimization.

Each Aitken Cascade Lake node contains two Cascade Lake processors and uses a dual single-port 100 Gbits/s Enhanced Data Rate (EDR) Mezzanine card to connect outward (see Inter-node Network). The use of four 200 Gbits/s High Data Rate (HDR) switches per enclosure forms the MPI and I/O InfiniBand fabrics within the Aitken cluster. Connecting the Aitken cluster to the Pleiades filesystems relies on additional sets of HDR switches and cables.

Instruction Sets

In addition to the instruction sets SSE, SSE2, SSE3, Supplemental SSE3, SSE4.1, SSE4.2, AVX, AVX2, AVX-512, and AVX512[F,CD,BW,DQ,VL], which are available in its Skylake predecessor, Cascade Lake also includes the new AVX-512 Vector Neural Network Instructions (VNNI), which provide significant, more efficient deep-learning inference acceleration. Cascade Lake also introduces in-hardware mitigations for the Spectre and Meltdown security flaws.

With 512-bit floating-point vector registers and two floating-point functional units, each capable of Fused Multiply-Add (FMA), a Cascade Lake core can deliver 32 double-precision floating-point operations per cycle.

Use the Intel compiler flag **`-xCORE-AVX512`** for Skylake and Cascade Lake-SP specific optimizations. The optimization flag **`-qopt-zmm-usage=high -xCORE-AVX512`** may benefit floating-point heavy applications running on Skylake and Cascade Lake.

Tip: If you want a single executable that will run on any of the Aitken, Electra, and Pleiades processor types, with suitable optimization to be determined at run time, you can compile your application using the option:
`-O3 -ipo -axCORE-AVX512,CORE-AVX2 -xAVX`.

Hyperthreading

Hyperthreading is turned ON.

Turbo Boost

Turbo Boost is turned ON. Maximum Turbo Frequency is 3.90 GHz for non-AVX, and 3.8 GHz for AVX2 and AVX-512.

Cache Hierarchy

The cache hierarchy of Cascade Lake is as follows:

- L1 instruction cache: 32 KB, private to each core; 64 sets; 64 B/line; 8-way
- L1 data cache: 32 KB, private to each core; 64 sets; 64 B/line; 8-way; fastest latency: 4 cycles; 128 B/cycle load bandwidth; 64 B/cycle store bandwidth; write-back policy
- L2 cache: 1 MB, private to each core; 64 B/line; 16-way; fastest latency: 14 cycles; 64 B/cycle bandwidth to L1 cache; write-back policy
- L3 cache: shared non-inclusive 1.375 MB/core; total of 27.5 MB, shared by 20 cores in each socket; 2,048 sets; 64 B/line; fully associative; fastest latency: 50 - 70 cycles; write-back policy

Memory Subsystem

Like Skylake, there are two sub-NUMA clusters in each Cascade Lake socket, creating two localization domains. There are three memory channels per sub-NUMA cluster. Each channel can be connected with up to two memory DIMMs. For the Aitken Cascade Lake configuration, there is one 16-gigabyte (GB) dual rank DDR4 DIMM with error correcting code (ECC) support per channel. In total, the amount of memory is 48 GB per sub-NUMA cluster, 96 GB per socket, and 192 GB per node.

The speed of each memory channel is increased from 2,666 MHz in Skylake to 2,933 MHz in Cascade Lake. An 8-byte read or write can take place per cycle per channel. With a total of six memory channels, the total half-duplex memory bandwidth is approximately 141 GB/s per socket.

On-chip Interconnect

The on-chip architecture of Cascade Lake SP uses the same mesh layout as that of Skylakeâ   where the cores and L3 caches are organized in rows and columnsâ   instead of the ring architecture used by earlier Xeon processors.

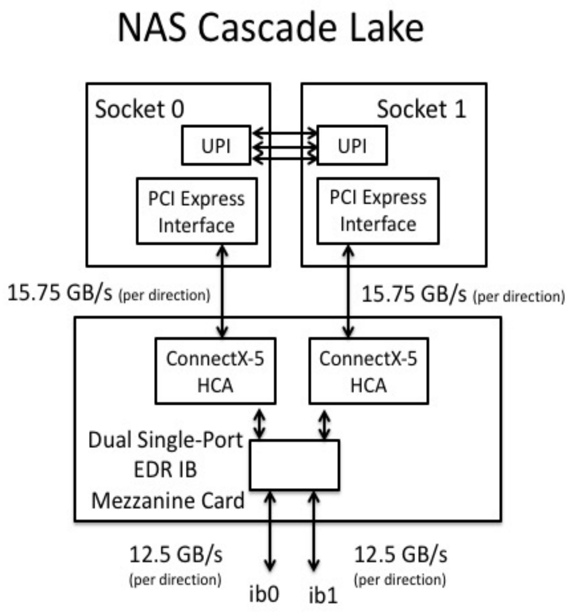
Inter-socket Interconnect

The Cascade Lake processors support up to three UPI links. The configuration at NAS has two links enabled. The UPI runs at a speed of 10.4 gigatransfers per second (GT/s). Each link contains separate lanes for the two directions. The total full-duplex bandwidth is 62.4 GB/s with three links and 41.6 GB/s with two links.

Inter-node Network

Like the network subsystems of the Pleiades Haswell and Broadwell nodes, and the Electra Broadwell and Skylake nodes, each Cascade Lake node uses two PCI Express interfaces (one from each socket) to connect to the Aitken and Pleiades InfiniBand (IB) fabrics. The use of PCIe 3.0 16-lane (x16) links enables a maximum bandwidth of 15.75 GB/s for each direction.

Like the Electra Skylake nodes, the Aitken Cascade Lake inter-node network makes use of the 100 Gbits/s EDR technology. As shown below, one PCIe is connected to the ib0 fabric via a single-port, four-lane (4X) EDR host channel adapter (HCA), in a dual single-port EDR IB mezzanine card, with an effective bandwidth of 100 Gbits/s (that is, 12.5 GB/s) for each direction. The other PCIe is connected to the ib1 fabric via another single-port 4x EDR HCA in the same mezzanine card.


NAS Cascade Lake
Socket 0
Socket 1
UPI
UPI
PCI Express Interface
PCI Express Interface
15.75 GB/s (per direction)
15.75 GB/s (per direction)
ConnectX-5 HCA
ConnectX-5 HCA
Dual Single-Port EDR IB Mezzanine Card
12.5 GB/s (per direction)
12.5 GB/s (per direction)
ib0
ib1

There are 1,152 Cascade Lake nodes in the Aitken cluster, which uses the liquid-cooled HPE 8600 system architecture. The nodes are partitioned as follows:

- Four E-cells (288 nodes per E-cell)
- Two compute racks and one cooling rack per E-cell (144 nodes per compute rack)
- Four enclosures (IRUs) per compute rack (36 nodes per enclosure)
- Nine compute trays (HPE XA730i "Saxon" blades) per enclosure (four nodes per tray)

The connection of the compute nodes relies on the use of four standard IB HDR switch blades per enclosure. Two of these switch blades facilitate the ib0 fabric, while the other two facilitate the ib1 fabric. Each switch blade has one 40-port ConnectX-6 ASIC with 18 ports connecting to the compute nodes, and 22 ports available for switch-to-switch links and connections to external targets. The connection of the 1,152 Cascade Lake nodes forms a 6-dimensional enhanced hypercube.

Notes:

- There are eight ASICs per enclosure in Electra Skylake, while there are four in Aitken Cascade Lake. This reduces a 1,152-node topology from seven dimensions in Electra Skylake to six dimensions in Aitken Cascade Lake.
- Only two out of the three UPI links are enabled.

References

- Intel Xeon Gold 6248 Processor
- Second Generation Intel Xeon Scalable Processors
- Intel 64 and IA-32 Architectures Optimization Reference Manual (Apr 2019 edition)
- HPE SGI 8600 QuickSpecs (Apr 15, 2019 edition)

Preparing to Run on Aitken Rome Nodes

To help you prepare for running jobs on Aitken's Rome nodes, this short user guide includes information on the general configuration of Rome nodes, compiling your code, and running PBS jobs.

Overview of Aitken Rome Nodes

Aitken includes 2,048 Rome nodes, which are partitioned into sixteen physical racks. Each node contains two 64-core AMD EPYC 7742 Rome sockets (2.25 GHz) and 512 GB of memory. The nodes are connected to the Aitken InfiniBand network (ib0 and ib1) via four-lane High Data Rate (4X HDR) devices and four-lane High Data Rate (4X HDR) switches for internode communication. The ib1 fabric is used mainly for I/O and is connected to the Pleiades Lustre filesystems. In addition, Aitken, Electra, and Pleiades share the same home filesystems, Pleiades front-end systems (PFEs), and PBS server. You can access Aitken only through PBS jobs submitted from the PFEs.

Operating System and Software Modules

The operating system on all Rome compute nodes is the Red Hat Enterprise Linux-based Tri-Lab Operating System Stack Version 3 (TOSS 3). Batch jobs use **`:aoe=toss3`** by default, so you do not need to add **`:aoe=toss3`** in your PBS resource request. The default $MODULEPATH includes:

- /usr/share/modules/modulefiles
- /nasa/modulefiles/toss3
- /nasa/modulefiles/spack/gcc-4.8/
- /nasa/modulefiles/pkgsrc/toss3/

Compilers for Rome Nodes

There are several compilers you can use to build your application:

- Intel Compilers:

  Some Intel compiler modules are available in the **`/nasa/modulefiles/toss3`** directory, however, several are in the **`/nasa/modulefiles/testing`** directory. To see what versions are available in these two directories, run:

  ```
  module use  nasamodulefilestesting
  module avail compintel
  ```

  Existing executables that are built with support for **`AVX512`** (for running on Skylake and Cascade Lake processors) will not run on the Rome nodes. Others that are built with support for **`AVX2`** may run on the Rome nodes; however, you should always check the result of your runs using existing executables. To rebuild your application using Intel compilers, use **`-march=core-avx2`**.

  Note: Although AMD recommends using **`-axCORE-AVX2`**, we do *not* recommend using it, as the resulting executable will run but will not actually take advantage of **`AVX2`** instructions. Executables built with **`-xCORE-AVX2`**, **`-xSSE4.2`**, etc., will not run on Rome processors.
- GNU Compilers:

  The default version of GNU Compiler Collection (GCC) under TOSS 3 (/usr/bin/gcc) is GCC 4.8.5. However, support for the Rome architecture is only available with GCC 9.x and later. There are several GCC versions in the /nasa/pkgsrc/toss3 directory that can be used with the Rome nodes. For example:

  ```
  module avail gcc
  -- /nasa/modulefiles/pkgsrc/toss3/ --
  gcc/10.2 gcc/10.3 gcc/7.5  gcc/9.3
  module load gcc/10.3
  ```

  The recommended compiler flag to use with GCC is **`-march=znver2`**.
- AMD Optimizing C/C++ Compilers (AOCC)

  To see available versions before loading an AOCC compiler, use the **`module avail comp-aocc`** command. For example:

  ```
  module avail comp-aocc
  -- /nasa/modulefiles/toss3 --
  comp-aocc/3.1.0

  module load comp-aocc/3.1.0
  ```

  The recommended compiler flag to use with AOCC is **`-march=znver2`**.
- PGI Compilers:

  Some PGI compiler modules are available in the **`/nasa/modulefiles/testing`** directory. There is no support specifically for Rome processors. If you want to compile your code using a PGI compiler, the recommended compiler flag is **`-tp=zen`**

For more compiler options, see Compiler Options Quick Reference Guide for AMD EPYC 7xx2 Series Processors.

MPI Library for Rome Nodes

HPE MPT is the only recommended MPI library for use on the AMD Rome nodes under TOSS 3. Use the **`mpi-hpe/mpt.2.25`** modulefile in the **`/nasa/modulefiles/toss3`** directory. Earlier versions such as **`mpi-hpe/mpt.2.23`** do not work properly for the Rome nodes.

```
module load mpiâ  hpe/mpt.2.25
```

Note: Although there is more than one version of MPT library installed on NAS systems, **`mpiâ  hpe/mpt.2.25`** is currently the only reliable version for running on the Rome nodes. You can access this version by loading **`mpiâ  hpe/mpt`**, which points to the recommended version and environmental settings:

```
% module load mpi-hpe/mpt
```

Running OpenMP or MPI/OpenMPI Hybrid Codes

Each Rome node has 128 physical cores. When Simultaneous Multi-Threading (SMT) is enabled in BIOS, there are 256 logical cores for use by OpenMP or MPI/OpenMPI hybrid codes.

To check whether SMT is enabled, run the following command on a Rome compute node:

```
 proccpuinfo   processor
```

If it shows 256, then SMT is enabled.

We recommend using the mbind.x tool for process and thread binding, as shown in the following example with four MPI processes and 32 OpenMP threads per process:

```
mpiexec n  uscicontoolsbinmbindx t  aout
```

If you use HPE MPT, you can also use the omplace tool for pinning MPI/OpenMP hybrid code.

Running PBS Jobs on Aitken Rome Nodes

To request Aitken Rome nodes, use **`:model=rom_ait`** in your PBS script, as shown in the example below.

The **`devel, normal, long, debug`**, and **`low`** queues are available for use with Rome nodes.

Note: The Standard Billing Unit (SBU) rate for Rome processors is 4.06.

Sample PBS Script For Aitken Rome Nodes

```
#PBS -l select=2:ncpus=128:mpiprocs=128:model=rom_ait
#PBS -l walltime=8:00:00
module load mpihpempt
module load compintel



mpiexec np  aout
```

For more information, see AMD Rome Processors.

Preparing to Run on Aitken Cascade Lake Nodes

To help you prepare for running jobs on Aitken's Cascade Lake compute nodes, this short user guide includes information on the general configuration of Cascade Lake nodes, compiling your code, running PBS jobs, and checking allocation usage.

Overview of Aitken Cascade Lake Nodes

Aitken includes 1,152 Cascade Lake nodes, which are partitioned into eight physical racks. Each node contains two 20-core Xeon Gold 6248 sockets (2.5 GHz) and 192 GB of memory. The nodes are connected to the Aitken InfiniBand network (ib0 and ib1) via four-lane Enhanced Data Rate (4X EDR) devices and four-lane High Data Rate (4X HDR) switches for internode communication. The ib1 fabric is used mainly for I/O and is connected to the Pleiades Lustre filesystems. In addition, Aitken, Electra, and Pleiades share the same home filesystems, Pleiades front-end systems (PFEs), and PBS server. You can access Aitken only through PBS jobs submitted from the PFEs.

Compiling Your Code For Cascade Lake Nodes

The Cascade Lake processors include the Advanced Vector Extensions 512 (AVX-512). Intel AVX-512 optimizations are included in Intel compiler version 16.0 and later versions. We recommend that you test the latest Intel compiler, through the command **`module load comp-intel/2020.4.304`**, which may include better optimizations for AVX-512.

For Cascade Lake-specific optimizations, use the compiler option **`-xCORE-AVX512`**.

If you want a single executable that will run on any of the Aitken, Electra, and Pleiades processor types, with suitable optimization to be determined at runtime, you can compile your application using the option:
`-O3 -axCORE-AVX512,CORE-AVX2 -xAVX`.

Note: The use of either of these options, **`-xCORE-AVX512`** or **`-axCORE-AVX512,`** could either improve or degrade performance of your code. Be sure to check performance with and without these flags before using them for production runs.

Running PBS Jobs on Aitken Cascade Lake Nodes

To request Aitken Cascade Lake nodes, use **`:model=cas_ait`** in your PBS script, as shown in the example below.

Note: Because MPT 2.15 and earlier versions do not support the ConnectX-5 host channel adapters (HCAs), the environment variables **`MPI_IB_XRC`** and **`MPI_XPMEM_ENABLED`** have been disabled for jobs running on Cascade Lake. If your MPI applications perform significant MPI collective operations and rely on having these two variables enabled to get good performance, use MPT 2.17 or newer versions. You can use the NAS-recommended MPT version with the command:
`module load mpi-hpe/mpt`

Sample PBS Script For Aitken Cascade Lake Nodes

```
#PBS -l select=10:ncpus=40:mpiprocs=40:model=cas_ait
#PBS -l walltime=8:00:00
#PBS -q normal
```

```
module load mpi-hpe/mpt
module load comp-intel/2020.4.304

cd $PBS_O_WORKDIR

mpiexec -np 400 ./a.out
```

For more information, see Cascade Lake Processors.

Electra

Electra Configuration Details

Electra, NASA's first prototype modular supercomputing system, is housed in a module a short distance from the primary NASA Advanced Supercomputing (NAS) facility. Electra is configured as follows:

- 16 Broadwell racks and 8 Skylake E-cells
- 3,456 nodes
- 124,416 cores
- 589 terabytes total memory
- 8.32 petaflops theoretical peak performance

Hostnames

There are 4 individual rack units (IRUs) in each rack. For every two racks, there is a rack leader controlling them. The naming convention for the nodes residing in every two racks uses only odd rack numbers.

Broadwell
: 18 nodes per IRU, with 144 nodes residing in every two racks (2 racks x 4 IRUs x 18 nodes).
: The hostnames for the Broadwell nodes are r[*x*]i[0-7]n[0-17], where *x* = odd numbers between 1 and 15.

Skylake
: 36 nodes per IRU, with 288 nodes residing in every two racks (2 racks x 4 IRUs x 36 nodes).
: The hostnames for the Skylake nodes are r[*x*]i[0-7]n[0-35], where *x* = odd numbers between 133 and 147.

Processor, Memory and Network Subsystems Statistics

Electra's system architecture is ICE X. The following table provides detailed configuration statistics for the processor, memory, and network subsystems:

	Processor
	Broadwell
CPU	14-Core Xeon E5-2680v4
Newest Instruction Set	AVX2
Hyperthreading	ON
TurboBoost	ON
CPU-Clock	2.4 GHz
Maximum Double Precision Floating Point Operations per Cycle per Core	16
# of Cores/node	28
Total # of Nodes	1,152
Total # of Cores	32,256

Total Double Precision TFlops	1,239
	Memory
L1 Cache	Local to each core; Instruction cache: 32K Data cache: 32K; Associativity: 8; Cache line size: 64B

L2 Cache	256 KB per core;
	Associativity: 8;
	Cache line size: 64B

L3 Cache	35 MB shared inclusive by the 14 cores; Associativity: 20; Cache line size: 64B
TLB	Local to each core
Default Page Size	4 KB
Memory/Core	4.6 GB; DDR4
Total Memory/node	128 GB;

Memory Speed and Bandwidth	2400 MHz;
	4 channels;
	76.8 GB/sec read/write

Intersocket Interconnect	QuickPath Interconnect 4.8 GHz, 9.6 GT/s, or 38.4 GB/sec
	Inter-node Network
IB Device on node	Dual single-port 4x FDR IB Mezzanine card (2 single-port HCAs); 56 Gbits/s
IB Switches between nodes	4x FDR; 56 Gbits/s

Preparing to Run on Electra Skylake Nodes

To help you prepare for running jobs on Electra's Skylake compute nodes, this short user guide includes information on the general configuration of Skylake nodes, compiling your code, running PBS jobs, and checking allocation usage.

Overview of Electra Skylake Nodes

Electra includes 2,304 Skylake nodes, which are partitioned into sixteen physical racks. Each node contains two 20-core Xeon Gold 6148 sockets (2.4 GHz) and 192 GB of memory. The nodes are connected to the Electra InfiniBand network (ib0 and ib1) via four-lane Enhanced Data Rate (4X EDR) devices and switches for internode communication. The ib1 fabric is used mainly for I/O and is connected to the Pleiades Lustre filesystems. In addition, Electra and Pleiades share the same home filesystems, Pleiades front-end systems (PFEs), and PBS server. You can access Electra only through PBS jobs submitted from the PFEs.

Usage of Electra Skylake nodes is charged to your project's allocation on Pleiades at the rate of 6.36 Standard Billing Units (SBUs).

Compiling Your Code For Skylake Nodes

The Skylake processors include the Advanced Vector Extensions 512 (AVX-512). Intel AVX-512 optimizations are included in Intel compiler version 16.0 and later versions. We recommend that you test the latest Intel compiler, **`2020.4.304`**, which may include better optimizations for AVX-512.

For Skylake-specific optimizations, use the compiler option **`-xCORE-AVX512`**.

If you want a single executable that will run on any of the Electra or Pleiades processor types, with suitable optimization to be determined at runtime, you can compile your application using the option:
`-O3 -axCORE-AVX512,CORE-AVX2 -xAVX`.

Note: The use of either of these options, **`-xCORE-AVX512`** or **`-axCORE-AVX512,`** could either improve or degrade performance of your code. Be sure to check performance with and without these flags before using them for production runs.

Running PBS Jobs on Electra Skylake Nodes

To request Electra Skylake nodes, use **`:model=sky_ele`** in your PBS script, as shown in the example below.

Important: Because MPT 2.15 and earlier versions do not support the ConnectX-5 host channel adapters (HCAs), the environment variables **`MPI_IB_XRC`** and **`MPI_XPMEM_ENABLED`** have been disabled for jobs running on Skylake. If your MPI applications perform significant MPI collective operations and rely on having these two variables enabled to get good performance, use MPT 2.16 or newer versions. To use the NAS-recommended MPT version, use the command:
`module load mpi-hpe/mpt`

Sample PBS Script For Electra Skylake Nodes

```
#PBS -l select=10:ncpus=40:mpiprocs=40:model=sky_ele
#PBS -l walltime=8:00:00
#PBS -q normal

module load mpi-hpe/mpt
module load comp-intel/2020.4.304

cd $PBS_O_WORKDIR

mpiexec -np 400 ./a.out
```

Checking Allocation Usage For Electra Skylake Jobs

To track allocation usage for jobs running on Electra Skylake nodes, run:

```
 acct_query -c electra_S
```

Examples:

- To track the total SBUs usage for one of your GIDs (for example, s0001) since 9/30/17:

  ```
  % acct_query -c electra_S -ps0001 -b 9/30/17
  ```
- To track the SBUs usage for each of your jobs run today:

  ```
  % acct_query -c electra_S -olow
  ```

For more information, see Skylake Processors.

Preparing to Run on Electra Broadwell Nodes

To help you prepare for running jobs on Electra's Broadwell compute nodes, this short review includes information on the general configuration, compiling your code, running PBS jobs, and checking allocation usage.

Overview of Electra Broadwell Nodes

Electra is housed in a module outside of the primary NAS facility. The system has 16 Broadwell racks, each containing 72 nodes. Each of the 72 nodes contains two 14-core E5-2680v4 (2.4 GHz) processor chips and 128 GB of memory.

Electra's InfiniBand fabric (ib1) is used mainly for I/O and is connected to the Pleiades Lustre filesystems. In addition, Electra and Pleiades share the same home filesystems, Pleiades front-end systems (PFEs), and PBS server. You can access Electra only through PBS jobs submitted from the PFEs.

Usage of Electra Broadwell nodes is charged to your project's allocation on Pleiades at the same Standard Billing Unit (SBU) rate as the Pleiades Broadwell nodes.

Compiling Your Code For Broadwell Nodes

See the **Compiling Your Code For Broadwell Nodes** section in Preparing to Run on Pleiades Broadwell Nodes.

Running PBS Jobs on Broadwell Nodes

To request Broadwell nodes, use **`model=bro_ele`** in your PBS script.

Note: If your job requests only **`model=bro_ele`** nodes, then by default PBS will run your job on either Pleiades or Electra Broadwell nodes, whichever becomes available first. If you specifically want your job to only run on Electra, then add **`-l site=static_broadwell`** to your job request. For example:

```
#PBS -l select=10:ncpus=28:mpiprocs=28:model=bro_ele
#PBS -l site=static_broadwell
```

Similarly, a request to use Pleiades Broadwell nodes using **`model=bro`** will by default run on either Pleiades or Electra Broadwell nodes. For more information, see Preparing to Run on Pleiades Broadwell Nodes.

Sample PBS Script For Broadwell Nodes

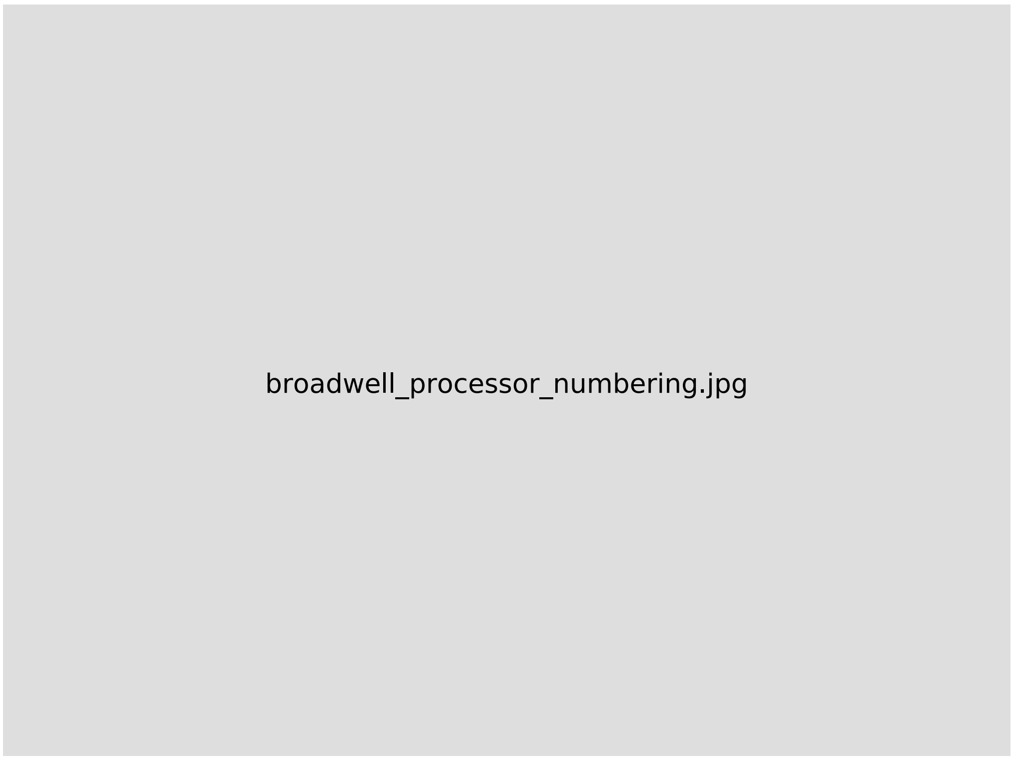
```
#PBS -l select=10:ncpus=28:mpiprocs=28:model=bro_ele
#PBS -l walltime=8:00:00
#PBS -q normal
module load comp-intel/2016.2.181 mpi-hpe/mpt
cd $PBS_O_WORKDIR
mpiexec -np 280 ./a.out
```

For more information about Broadwell nodes, see:

- Electra Configuration Details

- Broadwell Processors

Broadwell Processors



Microarchitecture

The Intel Broadwell processor incorporated into the Pleiades cluster is the 14-core E5-2680v4 model with a clock speed of 2.4 GHz. As the 14 nanometer (nm) die shrink of the Haswell microarchitecture, the Broadwell processor uses less power and is more efficient than the Haswell processor.

Instruction Sets

Like the Haswell processor, Broadwell supports single instruction, multiple data (SIMD) instruction sets, including several generations of Streaming SIMD Extensions (SSE, SSE2, SSE3, Supplemental SSE3, and SSE4), Advanced Encryption Standard (AES), and Advanced Vector Extensions (AVX and AVX2). Broadwell also supports some new instruction sets, including the Multi-Precision Add-Carry Instruction Extensions (ADX) for arbitrary-precision integer operations.

For information about AVX2 features and compiler support, see Haswell Processors.

Hyperthreading

Hyperthreading is turned ON.

Turbo Boost

Turbo Boost is turned ON.

Memory Subsystems

The memory hierarchy of Broadwell is as follows:

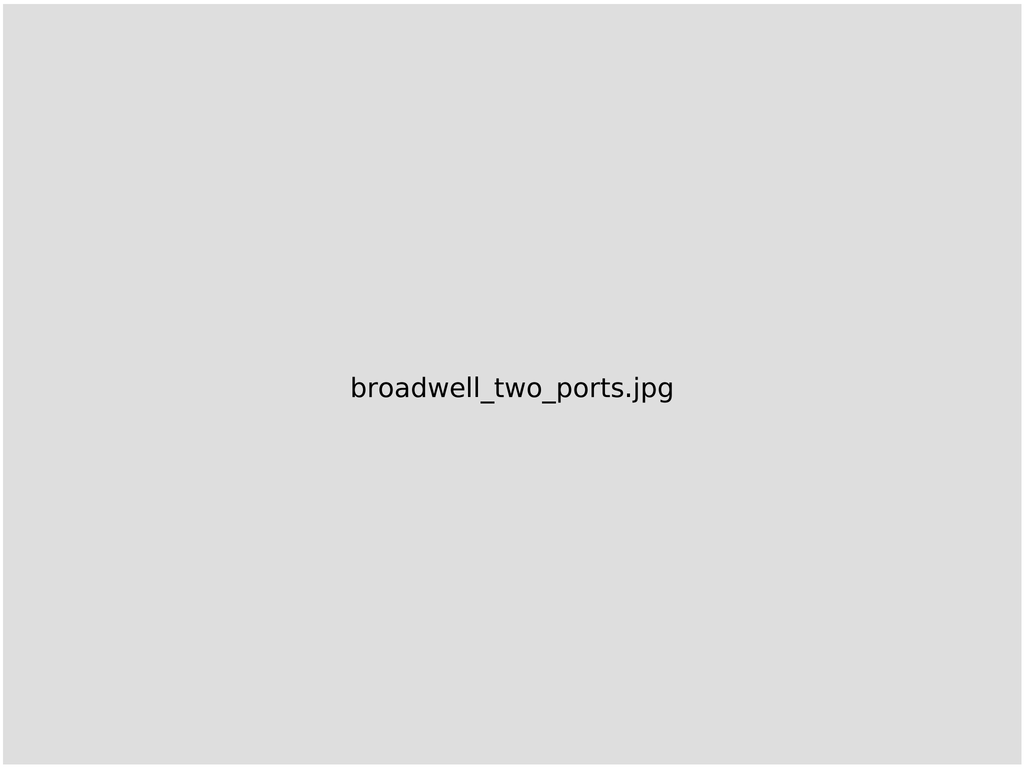
- L1 instruction cache: 32 KB, private to each core
- L1 data cache: 32 KB, private to each core
- L2 cache: 256 KB, private to each core
- L3 cache: 35 MB, shared by 14 cores in each socket
- Memory: 64 GB per socket, total of 128 GB per node

The Broadwell nodes are equipped with 2,400 MHz DDR4 memory to provide higher memory bandwidth. There are four memory channels per socket. Each channel can be connected with a maximum of two memory DIMMs. Of the eight memory DIMM slots for each socket, four are populated with 16-GB error correcting code (ECC)-registered DDR4 memory, for a total of 64 GB per socket. With two sockets in a node, the total memory per node is 128 GB.

Connecting the two sockets are two Intel QPI links running at a speed of 9.6 gigatransfers per second (GT/s). Each link contains separate lanes for the two directions. The total bandwidth (2 links x 2 directions) is 38.4 GB/sec.

Network Subsystem

The network subsystem of the Broadwell nodes is the same as that of the Haswell nodes, as shown in the following diagram. Each Broadwell node is equipped with two PCI Express (PCIe) interfaces (one from each socket). One PCIe interface is connected to the ib0 InfiniBand (IB) fabric via a single-port, four-lane, Fourteen Data Rate (4X FDR) host channel adapter (HCA), in a dual single-port FDR IB mezzanine card. The other PCIe interface is connected to the ib1 fabric via another single-port 4x FDR HCA in the same mezzanine card.



Skylake Processors

skylake_processor_numbering.jpg

The Intel Skylake processor incorporated into the Electra cluster is the 20-core Xeon Gold 6148 model with a base clock speed of 2.4 GHz. Skylake is a microarchitecture redesign using the same 14-nanometer (nm) manufacturing process technology as Broadwell, with multiple new features and enhancements in the on-chip and intersocket interconnects, memory, cache, and CPU.

Each Electra Skylake node makes use of the 100 Gbits/s four-lane Enhanced Data Rate (4X EDR) technology to connect to the rest of the Pleiades/Electra InfiniBand network.

Each small square in the diagram above represents a combination of a physical core and a L3 cache slice. For each physical core, there are two logical core labelings obtained from the "processor" entry of the `cat /proc/cpuinfo` output.

On-chip and Inter-socket Interconnects

In previous generations of Intel Xeon processors (such as Sandy Bridge, Ivy Bridge, Haswell, and Broadwell), the cores, L3 cache, memory controller, I/O controller and intersocket interconnect ports are connected with a on-chip ring architecture. This has the drawback of increased latency and bandwidth constraints when the number of cores per socket increases. To mitigate this issue, Skylake utilizes a mesh architecture that encompasses an array of vertical and horizontal paths, allowing communication from one core to another through a shortest path. Furthermore, each core and L3 slice has a combined Caching and Home Agent (CHA) that handles address mapping and information routing, and provides scalability of resources across the mesh.

The two sockets are connected with two Intel Ultra Path Interconnect (UPI) links, which deliver increased bandwidth and performance over the Intel Quick Path Interconnect (QPI) that was used in previous generations of Intel Xeon processors. The UPI runs at a speed of 10.4 gigatransfers per second (GT/s). Each link contains separate lanes for the two directions. The total full-duplex bandwidth (2 links x 2 directions) is 41.6 gigabytes per second (GB/s).

Note: With full-duplex communication between two components, both ends can transmit and receive information between each other simultaneously. With half-duplex communication, the transmission and reception must happen alternatively.

Memory Subsystem

There are two sub-NUMA clusters in each socket, creating two localization domains. Each domain contains one memory controller and ten L3 cache slices. Processes observe lower L3 and memory latency if they access data mapped to the L3 or memory in the same domain where the processes are running, compared with outside of the domain.

There are three memory channels per sub-NUMA cluster. Each channel can be connected with up to two memory DIMMs. For the Electra Skylake configuration, there is one 16-GB dual rank DDR4 DIMM with error correcting code (ECC) support per channel. In total, the amount of memory is 48 GB per sub-NUMA cluster, 96 GB per socket, and 192 GB per node.

The activation or deactivation of the sub-NUMA domains is controlled in the Basic Input/Output System (BIOS) firmware. Currently, they are deactivated for the Electra Skylake nodes.

The speed of each memory channel is 2,666 MHz. One 8-byte read or write can take place per cycle per channel. With a total of 6 memory channels, the total half-duplex memory bandwidth is approximately 128 GB/s per socket.

Cache Hierarchy

The cache hierarchy of Skylake is as follows:

- L1 instruction cache: 32 KB, private to each core; 64 B/line; 8-way
- L1 data cache: 32 KB, private to each core; 64 B/line; 8-way; fastest latency: 4 cycles
- L2 cache: 1 MB, private to each core; ; 64 B/line; 16-way; fastest latency: 12 cycles
- L3 cache: shared non-inclusive 1.375 MB/core; total of 27.5 MB, shared by 20 cores in each socket; fully associative; fastest latency: 44 cycles

In Broadwell, the L2 cache is 256 KB per core and the L3 cache is a shared inclusive cache with 2.5 MB per core. In Skylake, the cache hierarchy has changed to provide a larger L2 cache of 1 MB per core and a smaller shared non-inclusive 1.375 MB L3 cache per core.

Note: An inclusive L3 cache guarantees that every block that exists in the L2 cache also exists in the L3 cache. A non-inclusive L3 cache does not guarantee this.

A larger L2 cache increases the hit rate into the L2 cache, resulting in lower effective memory latency and lower demand on the mesh interconnect and L3 cache.

If the processor has a miss on all the levels of the cache, it fetches the line from memory and puts it directly into the L2 cache of the requesting core, rather than putting a copy into both the

L2 and L3 caches, as is done on Broadwell. When the cache line is evicted from the L2 cache, it is placed into L3 if it is expected to be reused.

Due to the non-inclusive nature of the L3 cache, the absence of a cache line in L3 does not indicate that the line is absent in private caches of any of the cores. Therefore, a snoop filter is used to keep track of the location of cache lines in the L1 or L2 caches of cores when a cache line is not allocated in L3. On the previous-generation processors, the shared L3 itself takes care of this task.

Processor

Like the Broadwell processors, Skylake supports single instruction, multiple data (SIMD) instruction sets, including several generations of Streaming SIMD Extensions (SSE, SSE2, SSE3, Supplemental SSE3, SSE4.1 and SSE4.2), and Advanced Vector Extensions (AVX and AVX2). In addition, it also includes MPX (Memory Protection Extensions), Intel SGX (Software Guard Extensions) and Advanced Vector Extensions 512 (AVX-512).

AVX-512 was originally introduced with the Intel Xeon Phi processors. Of the multiple AVX-512 instruction groups, Skylake comes with the AVX512F, AVX512CD, AVX512BW, AVX512DQ groups and a new AVX512VL feature.

With AVX-512, programs can pack eight double precision or 16 single precision floating-point numbers, or eight 64-bit integers, or 16 32-bit integers within the 512-bit vectors. With 512-bit floating-point vector registers and two floating-point functional units, each capable of Fused Multiply-Add (FMA), a Skylake core can deliver 32 floating-point operations per cycleâ double the number of operations of a Haswell/Broadwell core, or quadruple that of a Sandy Bridge/Ivy Bridge core can deliver.

Unlike SSE and AVX, which cannot be mixed without performance penalties, the mixing of AVX and Intel AVX-512 instructions is supported without penalty. AVX registers YMM0-YMM15 map into the Intel AVX-512 registers ZMM0-ZMM15, very much like SSE registers map into AVX registers. Therefore, in processors with Intel AVX-512 support, AVX and AVX2 instructions operate on the lower 128 or 256 bits of the first 16 ZMM registers.

Intel AVX-512 optimizations are included in Intel compiler version 16.0 and later versions. For Skylake-specific optimizations, use **`-xCORE-AVX512`**. With Intel compiler 2018 versions, using the optimization flag **`-qopt-zmm-usage=high -xCORE-AVX512`** may benefit floating-point heavy applications running on Skylake.

TIP: If you want a single executable that will run on any of the Electra or Pleiades processor types, with suitable optimization to be determined at run time, you can compile your application using the option **`-O3 -axCORE-AVX512 -xSSE4.2`**.

Hyperthreading

Hyperthreading is turned ON.

Turbo Boost

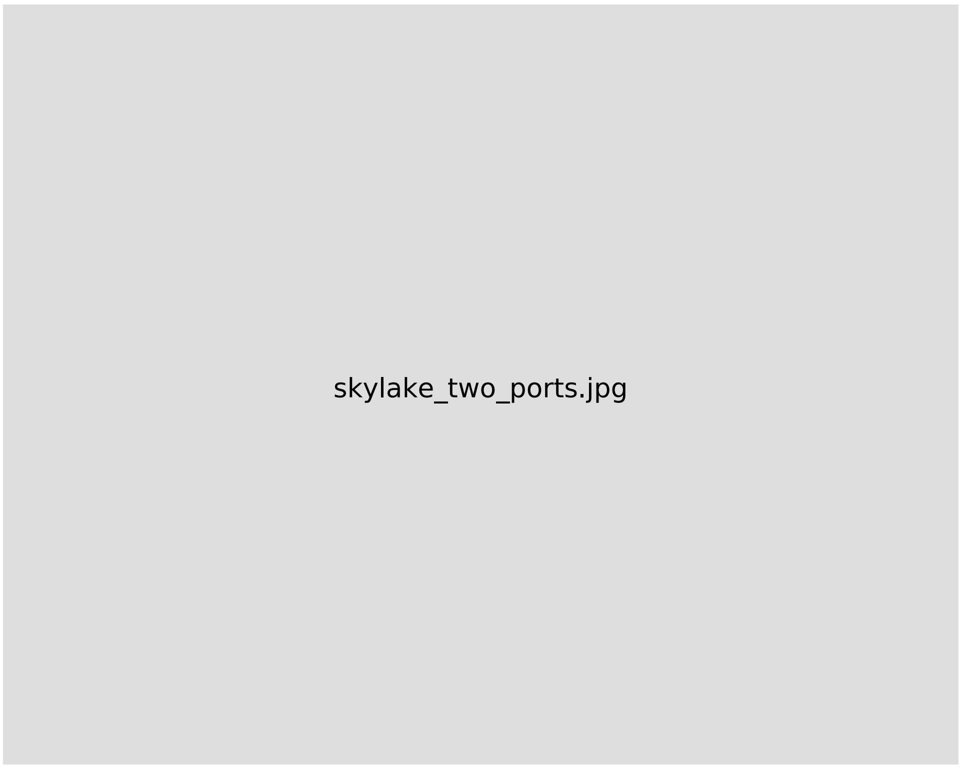
Turbo Boost is turned ON. Maximum Turbo Frequency is 3.70 GHz.

Inter-Node Network

Like the network subsystem of the Pleiades Haswell and Broadwell nodes, each Skylake node uses two PCI Express interfaces (one from each socket) to connect to the Pleiades and Electra Infiniband fabric. The use of PCIe 3.0 16-lane (x16) links enables a maximum bandwidth of 15.75 GB/s for each direction.

One enhancement in the NAS Skylake inter-node network is that it makes use of the 100 Gbits/s Enhanced Data Rate technology instead of the 56 Gbits/s Fourteen Data Rate technology used for the NAS Pleiades nodes (Sandy Bridge, Ivy Bridge, Haswell and Broadwell nodes). As shown below, one PCIe is connected to the ib0 Infiniband fabric via a single-port, four-lane, Enhanced Data Rate (4X EDR) host channel adapter (HCA), in a dual single-port EDR IB mezzanine card, with an effective bandwidth of 100 Gbits/s (that is 12.5 GB/s) for each direction. The other PCIe is connected to the ib1 fabric via another single-port 4x EDR HCA in the same mezzanine card.

Note: For nodes labeled with r[x]i[x]n[9-17,27-35], socket 0 is connected to ib0 and socket 1 is connected to ib1. For nodes labeled with r[x]i[x]n[0-8,18-26], socket 0 is connected to ib1 and socket 1 is connected to ib0.



There are 2,304 Skylake nodes in the Electra cluster. They are partitioned as follows:

- Eight E-cells (288 nodes per E-cell)
- Two compute racks and one cooling rack per E-cell (144 nodes per rack)
- Four individual rack units (IRUs) per compute rack (36 nodes per IRU)
- Nine compute blade (ICE XA SAXON blade) per IRU (4 nodes per blade)

The connection of the compute nodes relies on the use of four premium IB EDR switch blades per IRU. Each switch blade has two 36-port ConnectX-5 ASICs with 9 ports from each ASIC connecting to compute nodes and 18 ports from each ASIC for connecting to external targets. The connection of the 2,304 Skylake nodes forms an 8-dimensional hypercube. The topology of Electra (including both the Skylake and Broadwell nodes) is a 9-dimensional hypercube.

MPT 2.15 and older versions do not support the ConnectX-5 HCAs. To avoid jobs failing when using MPT 2.15 and older versions, the environment variables **`MPI_IB_XRC`** and **`MPI_XPMEM_ENABLED`** have been disabled for jobs running on Skylake. If your MPI applications perform significant MPI collective operations and rely on having these two variables enabled to get good performance, use MPT 2.16 or the forthcoming newer versions.

References

- Intel Xeon Processor Scalable Family Technical Overview
- Intel 64 and IA-32 Architectures Optimization Reference Manual
- HPE SGI 8600 QuickSpecs

Pleiades

Pleiades Configuration Details

Pleiades Hardware Summary

- 153 racks

- 10,872 nodes
- 236,032 CPU cores + 184,320 GPU cores
- 893 TB total memory
- 7.24 petaflops (PF) theoretical peak performance (CPU) + 0.275 PF (GPU)

Compute Node Hostnames

There are 4 individual rack units (IRUs) in each rack, with 18 compute nodes per IRU. The hostname of each node is based on the physical rack (r) and IRU (i) it resides in, and its node position in the IRU (n). For example: r301i2n3.

Note: A single rack leader controls two racks, so the naming convention for the 144 nodes residing in every two racks (2 racks x 4 IRUs x 18 nodes) uses only odd rack numbers.

The following naming conventions are used for the compute node hostnames:

Sandy Bridge
    r[3xx]i[0-7]n[0-17], where 3xx are *odd* numbers between 305-312 and 317-330;
    r313i[0-3]n[0-15] (GPU)
Ivy Bridge
    r[4xx]i[0-7]n[0-17], where 4xx are *odd* numbers between 401-472; and 481-482;
    483i[0-3]n[0-17]
Haswell
    r[5xx]i[0-7]n[0-17], where 5xx are *odd* numbers between 509-516, 573-580, and
    585-596; 583i[4-7]n[0-17]
Broadwell
    r[6xx]i[0-7]n[0-17], where 6xx are *odd* numbers between 601-608, and 617-636

Note: Pleiades compute nodes are accessible only through PBS jobs.

Pleiades Processor, Memory, and Network Subsystems Statistics

The following tables provide detailed configuration statistics for each type of node.

	Node Details		
	Sandy Bridge	**Ivy Bridge**	**Haswell**
Architecture	ICE X	ICE X	ICE X
Processor	8-core Xeon E5-2670	10-core Xeon E5-2680v2	12-core Xeon E5-2680
Newest Instruction Set	AVX	AVX	AVX2
Hyperthreading	ON	ON	ON

Turbo Boost	ON	ON	ON
CPU-Clock	2.6 GHz	2.8 GHz	2.5 GHz
Maximum Double Precision Flops/Cycle/Core	8	8	16
# of Cores/Node	16	20	24
Total # of Nodes	1,512	5,256	2,052
Total # of Cores	24,192	105,120	49,248
Total Double Precision TFlops	599	2,355	1,970

Memory

	Sandy Bridge	**Ivy Bridge**	**Haswell**
L1 Cache	Local to each core; Instruction cache: 32K Data cache: 32K; Associativity: 8; Cache line size: 64B	Local to each core; Instruction cache: 32K Data cache: 32K; Associativity: 8; Cache line size: 64B	Local to each core; Instruction cache: 3 Data cache: 32K; Associativity: 8; Cache line size: 64

L2 Cache	256 KB per core;
	Associativity: 8;
	Cache line size: 64B
		256 KB per core;
		Associativity: 8;
		Cache line size: 64B
			256 KB per core;
			Associativity: 8;
			Cache line size: 64

L3 Cache	20 MB shared by
	the 8 cores;
	Associativity: 20;
	Cache line size: 64B
		25 MB shared by
		the 10 cores;
		Associativity: 20;
		Cache line size: 64B
			30 MB shared by
			the 12 cores;
			Associativity: 20;
			Cache line size: 64

TLB	Local to each core	Local to each core	Local to each core
Default Page Size	4 KB	4 KB	4 KB
Memory/Core	2 GB; DDR3	3.2 GB; DDR3	5.3 GB; DDR4
Total Memory per Node	32 GB	64 GB; 3 nodes at 128 GB	128 GB

Memory Speed and Bandwidth	1600 MHz;
	4 channels;
	51.2 GB/sec read/write
		1866 MHz;
		4 channels;
		59.7 GB/sec read/write
			2133 MHz;
			4 channels;
			68 GB/sec read/writ

QuickPath Interconnect	4.0 GHz, 8.0 GT/s, or 32 GB/sec	4.0 GHz, 8.0 GT/s, or 32 GB/sec	4.8 GHz, 9.6 GT/s, or 38.4 GB/sec

Inter-Node Network

	Sandy Bridge	**Ivy Bridge**	**Haswell**
IB Device on Node	Dual-port 4x FDR IB Mezzanine card (1 dual-port HCA); 56 Gbits/s	Dual-port 4x FDR IB Mezzanine card (1 dual-port HCA); 56 Gbits/s	Dual single-port 4x FD Mezzanine card (2 single-port HCAs); 56 Gbits/s
IB Switches Between Nodes	4x FDR; 56 Gbits/s	4x FDR; 56 Gbits/s	4x FDR; 56 Gbits/s

Preparing to Run on Pleiades Broadwell Nodes

To help you prepare for running jobs on Pleiades Broadwell compute nodes, this short review includes the general node configuration, tips on compiling your code, and PBS script examples.

Overview of Pleiades Broadwell Nodes

Each Pleiades Broadwell rack contains 72 nodes; each node contains two 14-core E5-2680v4 (2.4 GHz) processors and 128 GB of memory, providing approximately 4.6 GB per coreâ   slightly less than the ~5.3 GB/core provided by the Haswell nodes.

The Broadwell nodes are connected to the Pleiades InfiniBand network (ib0 and ib1) via four-lane Fourteen Data Rate (4X FDR) devices and switches for internode communication.

The home and Lustre /nobackup filesystems are accessible from the Broadwell nodes.

Compiling Your Code For Broadwell Nodes

Like the Haswell processors, the Broadwell processors support the Advanced Vector Extensions 2 (AVX2) instructions, in addition to AVX (introduced with Sandy Bridge processors), SSE4.2 (introduced with Nehalem processors), and earlier generations of SSE.

AVX2 supports floating point fused multiply-add, integer vector instructions extended to 256-bit, and vectorization gather support, among other features. Your application may be able to take advantage of AVX2, however, not all applications can make effective use of this set of instructions. We recommend that you use the latest Intel compiler, by using the command **`module load comp-intel/2020.4.304,`** and experiment with the following sets of compiler options before making production runs on the Broadwell nodes:

- **`-O2 -xCORE-AVX2`**
- **`-O3-xCORE-AVX2`**

These compiler options generate an executable that is optimized for running on Broadwell and Haswell, but cannot run on any of the older processor types. If you prefer to generate a single executable that will run on any of the existing Pleiades processor types (Sandy Bridge, Ivy Bridge, Haswell, Broadwell, Skylake, and Cascade Lake), try using one of the following sets of compiler options:

- **`-O2 (or -O3)`**
- **`-O2 (or -O3) -axCORE-AVX512,CORE-AVX2 -xAVX`**

The use of **`-axCORE-AVX512,CORE-AVX2`** allows the compiler to generate multiple code paths with suitable optimization to be determined at run time.

If your application does not show performance improvements with **`-xCORE-AVX2`** or **`-axCORE-AVX512,CORE-AVX2 -xAVX (`**as compared with just **`-o2`** or **`-o3`**) when running on Broadwell and Haswell nodes, it is more advantageous to use the executable built with just **`-o2`** or **`-o3`** for production runs on all Pleiades processor types.

TIP: You can add the compiler options **`-ip`** or **`-ipo`** to instruct the compiler to look for ways to better optimize and/or vectorize your code. Also, to generate a report on how well your code is vectorized, add the compiler flag **`-vec-report2`**.
Notes:

- If you have an MPI code, we strongly recommend that you load the **mpi-hpe/mpt** module, which always points to the NAS recommended MPT version.
- Ensure your jobs run correctly on Broadwell nodes before you start production work.

Running PBS Jobs on Broadwell Nodes

A PBS job running with a fixed number of processes or threads should use fewer Broadwell nodes than other types of Pleiades nodes, for two reasons: Broadwell nodes have more cores and more memory.

To request Broadwell nodes, use **model=bro** in your PBS script. For example:

```
 #PBS -l select=xx:ncpus=yy:model=bro
```

Note: If your job requests only **model=bro** nodes, then by default PBS will run your job on either Pleiades or Electra Broadwell nodes, whichever becomes available first. If you specifically want your job to only run on Pleiades, then add **-l site=static_broadwell** to your job request. For example:

```
#PBS -l select=10:ncpus=28:mpiprocs=28:model=bro
#PBS -l site=static_broadwell
```

Similarly, a request to use Electra Broadwell nodes using **model=bro_ele** will by default run on either Pleiades or Electra Broadwell nodes. For more information, see Preparing to Run on Electra Broadwell Nodes.

Cores per Node

There are 28 cores per Broadwell node compared to 24 cores per Haswell, 20 cores per Ivy Bridge, and 16 cores per Sandy Bridge.

For example, if you have previously run a 240-process job with 10 Haswell nodes, 12 Ivy Bridge nodes, or 15 Sandy Bridge nodes, you should request 9 Broadwell nodes (where the first node can run 16 processes while the remaining 8 nodes can run 28 processes each).

```
For Broadwell
#PBS -lselect=1:ncpus=16:mpiprocs=16:model=bro+8:ncpus=28:mpiprocs=28:model=bro

For Haswell
#PBS -lselect=10:ncpus=24:mpiprocs=24:model=has

For Ivy Bridge
#PBS -lselect=12:ncpus=20:mpiprocs=20:model=ivy

For Sandy Bridge
#PBS -lselect=15:ncpus=16:mpiprocs=16:model=san
```

Memory

Except for the Haswell node type, the Broadwell type provides more memory compared to the other Pleiades processor types both on a per node or per core basis.

For example, to run a job that needs 4 GB of memory per process, you can fit 7 processes on a 16-core Sandy Bridge node with ~30 GB/node, 15 processes on a 20-core Ivy Bridge node with ~60 GB/node, 24 processes on a 24-core Haswell node with ~122 GB/node, and 28 processes on a 28-core Broadwell node with ~122 GB/node.

Note: For all processor types, a small amount of memory per node is reserved for system usage. Therefore, the amount of memory available to a PBS job is slightly less than the total physical memory.

Sample PBS Script For Broadwell Nodes

```
#PBS -lselect=10:ncpus=28:mpiprocs=28:model=bro
#PBS -q devel
module load comp-intel/2020.4.304 mpi-hpe/mpt
cd $PBS_O_WORKDIR
mpiexec -np 280 ./a.out
```

For more information about Broadwell nodes, see:

- Pleiades Configuration Details
- Broadwell Processors

Preparing to Run on Pleiades Haswell Nodes

To help you prepare for running jobs on Pleiades Haswell compute nodes, this short review includes the general node configuration, tips on compiling your code, and PBS script examples.

Overview of Haswell Nodes

Pleiades has 29 Haswell racks, each comprising 72 nodes. Each node contains two 12-core E5-2680v3 (2.5 GHz) processor chips and 128 GB of memory, providing approximately 5.3 GB of memory per coreâ   the highest among all Pleiades processor types.

The Haswell nodes are connected to the Pleiades InfiniBand (ib0 and ib1) network via four-lane Fourteen Data Rate (4X FDR) devices and switches for inter-node communication.

The Pleiades home and Lustre (/nobackup) filesystems are accessible from the Haswell nodes.

Compiling Your Code For Haswell Nodes

The Haswell processors support the following sets of single instruction, multiple data (SIMD) instructions:

- Advanced Vector Extensions 2 (AVX2)
- AVX (introduced with Sandy Bridge processors)
- Streaming SIMD Extensions 4.2 (SSE 4.2; introduced with Nehalem processors)
- Earlier generations of SSE

The AVX2 instruction set, introduced with Haswell processors, includes the following new features:

- Floating-point fused multiply-add (FMA) support, which can double the number of peak floating-point operations compared with those run without FMA. With 256-bit floating-point vector registers and two floating-point functional units, each capable of FMA, a Haswell core can deliver 16 floating-point operationsâ   double the number of operations a Sandy Bridge or Ivy Bridge core can deliver.
- Integer-vector instructions support is extended from 128-bit to 256-bit capability.
- Gather vectorization support is enhanced to allow vector elements to be loaded from non-contiguous memory locations.

Your application may be able to take advantage of AVX2, however, not all applications can make effective use of this set of instructions. We recommend that you use the latest Intel compiler, through the command `module load comp-intel/2020.4.304` and experiment with the following sets of compiler options before making productions runs on the Haswell nodes.

To generate an executable that is optimized for running on Haswell and Broadwell nodes (but will not run on any other Pleiades processor types), use:

```
-O2 (or -O3) -xCORE-AVX2
```

To generate a single executable that will run on any of the existing Pleiades processor types (Sandy Bridge, Ivy Bridge, Haswell, Broadwell, Skylake, and Cascade Lake), use:

- **`-O2 (or -O3)`**
- **`-O2 (or -O3) -axCORE-AVX512,CORE-AVX -xAVX`**

Note: Using **`axCORE-AVX512,CORE-AVX2`** allows the compiler to generate multiple code paths with suitable optimization to be determined at run time.

If your application does not show performance improvements on Haswell using either of the two AVX2 options (**`-xCORE-AVX2`** or **`-axCORE-AVX512,CORE-AVX2 -xAVX`**), as compared with just **`-o2`** or **`-03`**, we recommend that you use the executable built with just **`-o2`** or **`-03`** for production runs on all Pleiades processor types. Executables built with AVX or AVX2 consume more power.

If you have an MPI code that uses the HPE MPT library, use the NAS Recommended MPT version, with the command:

```
module load mpi-hpe/mpt
```

TIP: If you include the **`-ipo`** compiler option to better optimize and/or vectorize your code, be aware that using the Intel compiler with any HPE/SGI MPT libraries can produce warning messages (regarding "**`unresolved cpuset_xxxx`**" and "**`bitmask_xxx`**" routines) during linking. These messages can be ignored. To make them stop, you can add the **`-lcpuset -lbitmask`** options.
Important: Ensure your jobs run correctly on Haswell nodes before starting production work.

Running PBS Jobs on Haswell Nodes

To request Haswell nodes, use **`:model=has`** in your PBS script. For example:

```
#PBS -l select=xx:ncpus=yy:model=has
```

A PBS job running with a fixed number of processes or threads should use fewer Haswell nodes than the older Pleiades processor types, for the following two reasons:

- There are 24 cores per Haswell node (compared with 20 cores per Ivy Bridge, and 16 cores per Sandy Bridge node).

  For example, if you have previously run a 240-process job on 12 Ivy Bridge nodes or 15 Sandy Bridge nodes, you would request 10 Haswell nodes instead:

  ```
  For Haswell
  #PBS -lselect=10:ncpus=24:mpiprocs=24:model=has

  For Ivy Bridge
  #PBS -lselect=12:ncpus=20:mpiprocs=20:model=ivy

  For Sandy Bridge
  #PBS -lselect=15:ncpus=16:mpiprocs=16:model=san
  ```
- Haswell processors provide more memory per core than the other Pleiades processor types. For example, to run a job that needs 5 GB of memory per process, you can fit the following number of processes on each type:
  - 24 processes on a 24-core Haswell node (or 28-core Broadwell node) with ~122 GB/node
  - 12 processes on a 20-core Ivy Bridge node with ~60 GB/node
  - 6 processes on a 16-core Sandy Bridge node with ~30 GB/node

Note: For all processor types, a small amount of memory per node is reserved for system usage. Therefore, the amount of memory available to a PBS job is slightly less than the total physical memory.

Sample PBS Script For Haswell Nodes

```
#PBS -lselect=10:ncpus=24:mpiprocs=24:model=has
```

```
#PBS -q devel

module load comp-intel/2020.4.304 mpi-hpe/mpt

cd $PBS_O_WORKDIR

mpiexec -np 240 ./a.out
```

For more detailed information about the Haswell nodes, see Haswell Processors.

Preparing to Run on Pleiades Ivy Bridge Nodes

To help you prepare for running jobs on Pleiades Ivy Bridge compute nodes, this short review includes the general node configuration, tips on compiling your code, and PBS script examples.

Overview of Ivy Bridge Nodes

Pleiades has 75 Ivy Bridge racks, each containing 72 nodes. Each node contains two 10-core E5-2680v2 (2.8 GHz) processor chips and 64 GB of memory, providing 3.2 GB of memory per core.

The Ivy Bridge nodes are connected to the Pleiades InfiniBand (ib0 and ib1) network via the four-lane Fourteen Data Rate (4X FDR) devices and switches for inter-node communication.

The Lustre filesystems, /nobackupp*X*, are accessible from the Ivy Bridge nodes.

Compiling Your Code For Ivy Bridge Nodes

Like the Sandy Bridge processor, the Ivy Bridge processor uses Advanced Vector Extensions (AVX), which is a set of instructions for doing Single Instruction Multiple Data (SIMD) operations on Intel architecture processors.

To take advantage of AVX, we recommend that you compile your code on Pleiades with an Intel compiler (version 12 or newer; use **`module load comp-intel/2020.4.304`** to get the latest version), using either of the following compiler flags:

- To run only on Sandy Bridge or Ivy Bridge processors: **`-O2 (or -O3) -xAVX`**
- To run on all Pleiades processor types (including Sandy Bridge, Ivy Bridge, Haswell, Broadwell, Skylake and Cascade Lake): **`-O2 (or -O3) -axCORE-AVX512,CORE-AVX2 -xAVX`**

You can also add the compiler options **`-ip`** or **`-ipo`**, which allow the compiler to look for ways to better optimize and/or vectorize your code.

To get a report on how well your code is vectorized, add the compiler flag **`-vec-report2`**.

If you have an MPI code that uses the HPE MPT library, use the NAS Recommended MPT version, with the command:

```
module load mpi-hpe/mpt
```

Note that mpt.2.04 and earlier versions do not support FDR.

TIP: Ensure that your jobs run correctly on Ivy Bridge nodes before starting production work.

Running PBS Jobs on Ivy Bridge Nodes

To request Ivy Bridge nodes, use **`:model=ivy`** in your PBS script:

```
#PBS -l select=xx:ncpus=yy:model=ivy
```

A PBS job running with a fixed number of processes or threads should use fewer Ivy Bridge nodes than Sandy Bridge nodes for two reasons:

- There are 20 cores per Ivy Bridge node, compared with 16 cores per Sandy Bridge node.

  For example, if you have previously run a 240-process job with 15 Sandy Bridge nodes, you should request 12 Ivy Bridge nodes instead.

  ```
  For Ivy Bridge
  #PBS -lselect=12:ncpus=20:mpiprocs=20:model=ivy

  For Sandy Bridge
  #PBS -lselect=15:ncpus=16:mpiprocs=16:model=san
  ```
- The Ivy Bridge processor provides more memory, per node and per core, than the Sandy Bridge processor type.

  For example, to run a job that needs 2.5 GB of memory per process, you can fit the following number of processes on each type:

  - 12 processes on a 16-core Sandy Bridge node with ~30 GB/node
  - 20 processes on a 20-core Ivy Bridge node with ~60 GB/node

  Note: For all processor types, a small amount of memory per node is reserved for system usage. Thus, the amount of memory available to a PBS job is slightly less than the total physical memory.

Sample PBS Script For Ivy Bridge

```
#PBS -lselect=12:ncpus=20:mpiprocs=20:model=ivy
#PBS -q normal
module load comp-intel/2020.4.304 mpi-hpe/mpt
cd $PBS_O_WORKDIR
mpiexec -np 240 ./a.out
```

For more information about Ivy Bridge nodes, see:

- Pleiades Configuration Details
- Ivy Bridge Processors

Preparing to Run on Pleiades Sandy Bridge Nodes

To help you prepare for running jobs on Pleiades Sandy Bridge compute nodes, this short review includes the general node configuration, tips on compiling your code, and PBS script examples.

Overview of Sandy Bridge Nodes

Pleiades has 26 Sandy Bridge racks, each containing 72 nodes. Each of the 72 nodes contains two 8-core E5-2670 processor chips and has 32 GB of memory.

The E5-2670 processor speed is 2.6 GHz when Turbo Boost is OFF, and can reach 3.3 GHz when Turbo Boost is ON. When Turbo Boost is enabled, idle cores are turned off and and power is channeled to the active cores, making them more efficient. The net effect is that the active cores perform above their clock speed (that is, overclocked).

The Sandy Bridge nodes are connected to the Pleiades InfiniBand (ib0 and ib1) network via the four-lane Fourteen Data Rate (4x FDR) devices and switches for inter-node communication.

The Lustre filesystems, /nobackupp*X*, are accessible from the Sandy Bridge nodes.

Compiling Your Code For Sandy Bridge Nodes

One important feature of the Sandy Bridge processor is its use of the Advanced Vector Extensions (AVX), which is a set of instructions for doing Single Instruction Multiple Data (SIMD) operations on Intel architecture processors.

AVX uses 256-bit floating point registers, which are twice as wide as the 128-bit registers used in the earlier processor models on Pleiades. With two floating-point functional units and 256-bit registers (which can hold four double-precision floating-point values or eight single precision floating point values), a code with well-vectorized loops can achieve a maximum of either eight double-precision floating-point operations (flops) per cycle, per core or 16 single-precision flops per cycle, per core.

To take advantage of AVX, we recommend that you compile your code with an Intel compiler (version 12 or newer; use **`module load comp-intel/2020.4.304`** to get the latest version) on Pleiades, using either of the following compiler flags:

- To run only on Sandy Bridge (or Ivy Bridge) processors: **`-O2 (or -O3) -xAVX`**
- To run on all Pleiades processor types (including Sandy Bridge, Ivy Bridge, Haswell, Broadwell, Skylake and Cascade Lake): **`-O2 (or -O3) -axCORE-AVX512,CORE-AVX2 -xAVX`**

You can also add the compiler options **`-ip`** or **`-ipo`**, which allow the compiler to look for ways to better optimize and/or vectorize your code.

To get a report on how well your code is vectorized, add the compiler flag **`-vec-report2`**.

To compare the performance differences between using AVX and not using AVX, we recommend that you create separate executables: one with **`-xAVX`** or **`-aCORE-AVX512,CORE-AVX2 -xAVX,`** and another one without them. If you do not notice much performance improvement using these flags, then your code does not benefit from AVX.

If you have an MPI code that uses the HPE MPT library, use the **`mpi-hpe/mpt`** module, which always points to the NAS recommended MPT version.

Note that mpt.2.04 and earlier versions do not support FDR.

TIP: Ensure that your jobs run correctly on Sandy Bridge nodes before you start production work.

Running PBS Jobs on Sandy Bridge Nodes

To request Sandy Bridge nodes, use **`:model=san`** in your PBS script:

```
#PBS -l select=xx:ncpus=yy:model=san
```

To request the devel queue, use either of the following methods:

- In your PBS script, add:

  ```
  #PBS -q devel
  ```
- In your **`qsub`** command line, use:

  ```
  pfe% qsub -q devel your_pbs_script
  ```

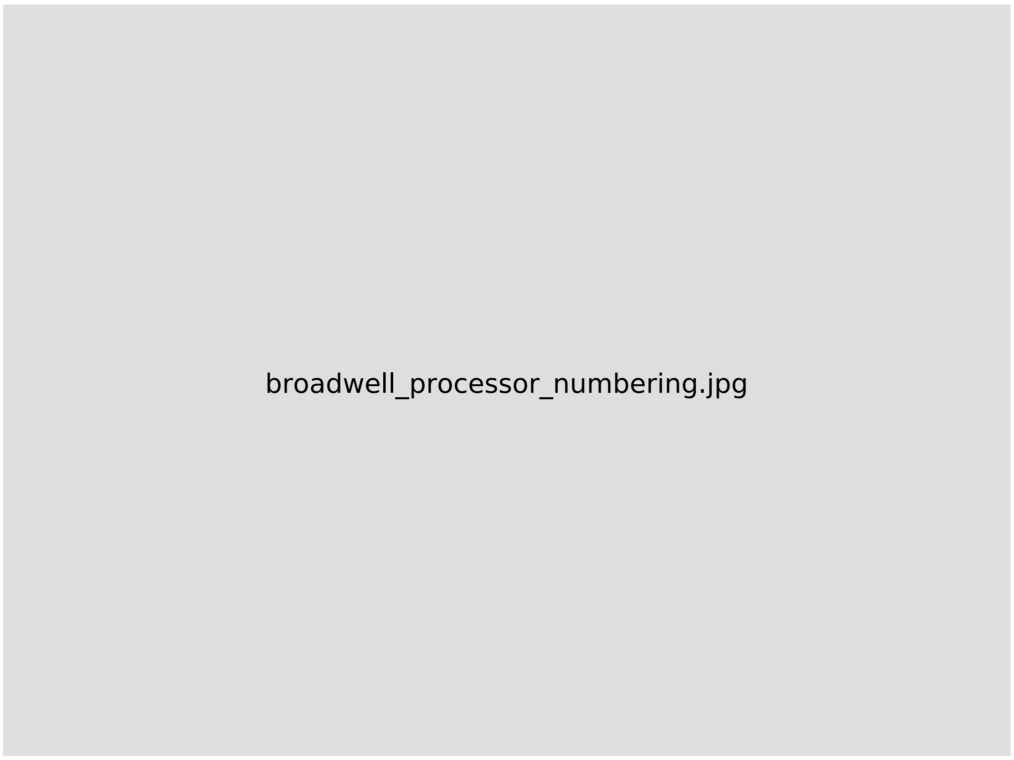
Sample PBS Script For Sandy Bridge

```
#PBS -lselect=3:ncpus=16:mpiprocs=16:model=san
#PBS -q devel

module load comp-intel/2020.4.304 mpi-hpe/mpt

cd $PBS_O_WORKDIR

mpiexec -np 48 ./a.out
```

For more information about Sandy Bridge nodes, see:

- Pleiades architecture overview
- Sandy Bridge Processors

Broadwell Processors



Microarchitecture

The Intel Broadwell processor incorporated into the Pleiades cluster is the 14-core E5-2680v4 model with a clock speed of 2.4 GHz. As the 14 nanometer (nm) die shrink of the Haswell microarchitecture, the Broadwell processor uses less power and is more efficient than the Haswell processor.

Instruction Sets

Like the Haswell processor, Broadwell supports single instruction, multiple data (SIMD) instruction sets, including several generations of Streaming SIMD Extensions (SSE, SSE2, SSE3, Supplemental SSE3, and SSE4), Advanced Encryption Standard (AES), and Advanced Vector Extensions (AVX and AVX2). Broadwell also supports some new instruction sets, including the Multi-Precision Add-Carry Instruction Extensions (ADX) for arbitrary-precision integer operations.

For information about AVX2 features and compiler support, see Haswell Processors.

Hyperthreading

Hyperthreading is turned ON.

Turbo Boost

Turbo Boost is turned ON.

Memory Subsystems

The memory hierarchy of Broadwell is as follows:

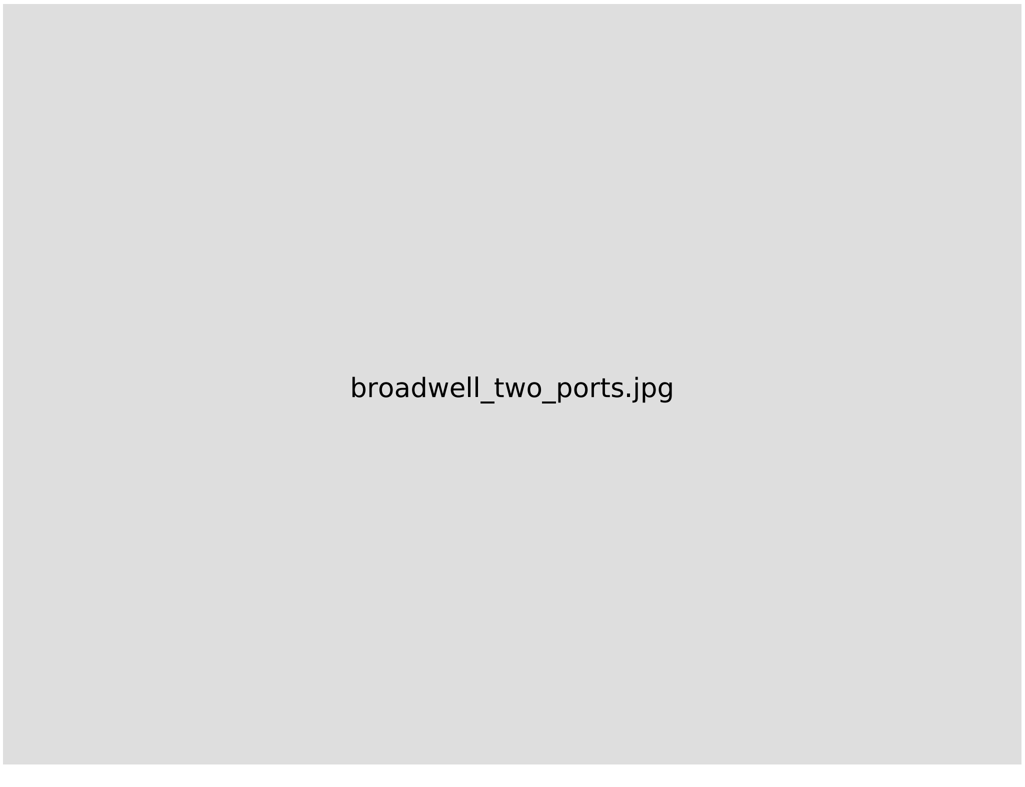
- L1 instruction cache: 32 KB, private to each core
- L1 data cache: 32 KB, private to each core
- L2 cache: 256 KB, private to each core
- L3 cache: 35 MB, shared by 14 cores in each socket
- Memory: 64 GB per socket, total of 128 GB per node

The Broadwell nodes are equipped with 2,400 MHz DDR4 memory to provide higher memory bandwidth. There are four memory channels per socket. Each channel can be connected with a maximum of two memory DIMMs. Of the eight memory DIMM slots for each socket, four are populated with 16-GB error correcting code (ECC)-registered DDR4 memory, for a total of 64 GB per socket. With two sockets in a node, the total memory per node is 128 GB.

Connecting the two sockets are two Intel QPI links running at a speed of 9.6 gigatransfers per second (GT/s). Each link contains separate lanes for the two directions. The total bandwidth (2 links x 2 directions) is 38.4 GB/sec.

Network Subsystem

The network subsystem of the Broadwell nodes is the same as that of the Haswell nodes, as shown in the following diagram. Each Broadwell node is equipped with two PCI Express (PCIe) interfaces (one from each socket). One PCIe interface is connected to the ib0 InfiniBand (IB) fabric via a single-port, four-lane, Fourteen Data Rate (4X FDR) host channel adapter (HCA), in a dual single-port FDR IB mezzanine card. The other PCIe interface is connected to the ib1 fabric via another single-port 4x FDR HCA in the same mezzanine card.



Haswell Processors

Microarchitecture

The Intel Haswell processor incorporated into the Pleiades supercomputer is the 12-core E5-2680v3 model. Haswell uses the 22 nanometer (nm) transistors that were introduced with Ivy Bridge processor, but has a newer microarchitecture that focuses on lower power consumption and higher efficiency.

Instruction Sets

Like the Sandy Bridge and Ivy Bridge processors, Haswell supports single instruction, multiple data (SIMD) instruction sets, including several generations of Streaming SIMD Extensions (SSE, SSE2, SSE3, Supplemental SSE3, and SSE4), Advanced Encryption Standard (AES), and Advanced Vector Extensions (AVX).

In addition, the AVX2 instruction set was introduced with Haswell processors.

Main Features of AVX2

Features include:

- Floating-point fused multiply-add (FMA) support, which can double the number of peak floating-point operations compared with those run without FMA. With 256-bit floating-point vector registers and two floating-point functional units, each capable of FMA, a Haswell core can deliver 16 floating-point operationsâ   double the number of operations a Sandy Bridge or Ivy Bridge core can deliver.
- Integer-vector instructions support has been extended from 128-bit to 256-bit capability.

- Gather vectorization support, enhanced to enable vector elements to be loaded from non-contiguous memory locations.

Compiler Support for AVX2

AVX2 is formally supported by Intel compilers starting with version 12.1. To take advantage of AVX2, use the following compiler and options:

Compiler

You can use `comp-intel/2012.0.032` or later modules. We recommend using `comp-intel/2015.0.090`.

Options

Use either `-xCORE-AVX2` or `-axCORE-AVX2`.

Both the `-xCORE-AVX2` and `-axCORE-AVX2` options turn on `-fma`, which computes `a x b + c` in one rounding. This provides higher accuracy than `-no-fma`, which computes `a x b + c` in two steps (first `a x b`, then `+ c`) and two roundings. Turn off FMA if you want your results to match legacy ones.

TIP: An application that is compiled with AVX2 instructions can run only on Haswell and Broadwell nodes. If you want a single executable that will run on any of the Pleiades processor types, with suitable optimization to be determined at run time, you can compile your application using the option `-O3 -ipo -axCORE-AVX2 -xSSE4.2`.
For Fortran codes, consider using the option `-align array32byte` to align your vector arrays on 32-byte boundaries for best performance and reliability. This option is available in Intel compiler version 13 (for example, comp-intel/2013.1.117 and later).

AVX2 is also supported in the GNU Compiler Collection (GCC) starting with version 4.7. Use compiler options such as `-march=core-avx-2` or `-mavx2`, `-mfma` if you want to use AVX2.

Hyperthreading

Hyperthreading is turned ON.

Turbo Boost

Turbo Boost is turned ON.

Memory Subsystems

The memory hierarchy of Haswell is as follows:

- L1 instruction cache: 32 KB, private to each core
- L1 data cache: 32 KB, private to each core
- L2 cache: 256 KB, private to each core
- L3 cache: 30 MB, shared by 12 cores in each socket
- Memory: 64 GB per socket, total of 128 GB per node

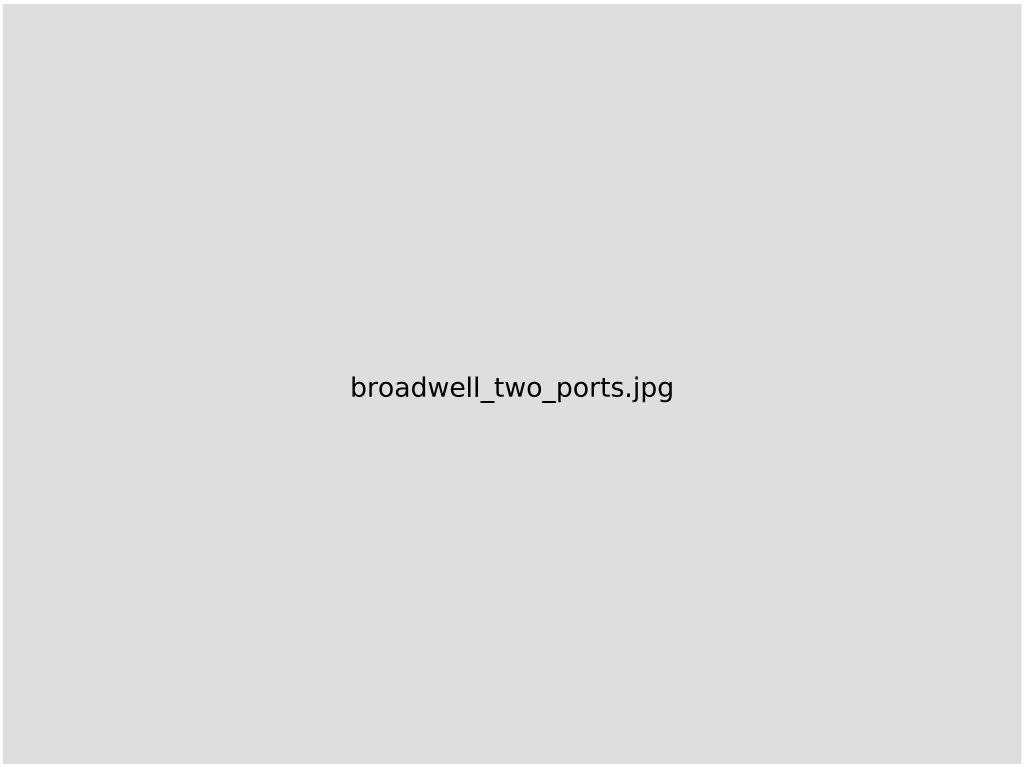
The Haswell nodes are equipped with higher speed (2,133 MHz) DDR4 memory to provide higher memory bandwidth. There are four 2,133 MHz memory channels per socket. Each channel can be connected with up to two memory DIMMs. Of the eight memory DIMM slots for each socket, four are populated with 16-GB error correcting code (ECC)-registered DDR4 memory, for a total of 64 GB per socket. With two sockets in a node, the total memory per node is 128 GB.

Connecting the two sockets are two Intel QPI links running at a speed of 9.6 gigatransfers per second (GT/s) . Each link contains separate lanes for the two directions. The total bandwidth (2 links x 2 directions) is 38.4 GB/sec.

Network Subsystem

The Haswell and Broadwell nodes are connected to the two fabrics (ib0 and ib1) of the Pleiades InfiniBand (IB) network in a different way from the earlier processor nodes, as shown in the diagram below. Note the following differences:

- In Haswell and Broadwell node connections, each of the two PCI Express Interfacesâ one from each socketâ is connected to a separate single-port, four-lane, Fourteen Data Rate (4X FDR) host channel adapter (HCA), in a dual single-port FDR IB mezzanine card.
- In Sandy Bridge and Ivy Bridge node connections, only one of the two PCI Express Interfaces is connected to a dual-port FDR IB HCA, in a dual-port FDR IB mezzanine card.



The IB mezzanine card sits on a "sister board" next to the motherboard on each node. The motherboard contains the two processor sockets. There are 18 nodes per individual rack unit. To join the ib0 fabric, the nodes are connected to two Mellanox FDR IB switches, in an ICE X IB Premium Blade. To join the ib1 fabric, another set of connections between the 18 nodes and a second Premium Blade is established.

Ivy Bridge Processors

Core Labeling

The core labeling in Ivy Bridge is contiguous. That is, cores 0-9 are in the first socket and cores 10-19 are in the second socket.

When using the HPE MPT library, the environment variable MPI_DSM_DISTRIBUTE is set to ON by default for the Ivy Bridge nodes.

Instruction Set

The Ivy Bridge processor is a die shrink (22 nm) of the Sandy Bridge processor (32 nm). It uses the Sandy Bridge micro-architecture, which contains Advanced Vector Extensions (AVX), a set of instructions for doing Single Instruction Multiple Data (SIMD) operations. These extensions widen the vector registers from 128 bits to 256 bits, so the floating-point hardware can sustain 16 single-precision or 8 double-precision floating point operations per cycle. As a result, even though the CPU clock speed of the Ivy Bridge processor (2.8 GHz) is lower than that of the earlier processor models on Pleiades, the floating-point performance can be higher for some applications.

AVX is formally supported in Intel compilers starting with version 12. Use the **`comp-intel/2012.0.032`** or newer modules (as of November 2014, **`comp-intel/2015.0.090`** is recommended) to take advantage of AVX. For Fortran codes, consider using the option **`-align array32byte`** (available in Intel compiler version 13 and above) to align your vector arrays on 32-byte boundaries for best performance and reliability.

AVX is also supported in the GNU Compiler Collection starting with version 4.6. An application that is compiled with AVX instructions can run only on Sandy Bridge or Ivy Bridge.

TIP: If you want a single executable that will run on any of the Pleiades processor types, with suitable optimization to be determined at run time, you can compile your application using the option **`-O3 -ipo -axCORE-AVX2 -xSSE4.2`**.

Hyperthreading

Hyperthreading is turned ON.

Turbo Boost

Turbo Boost is turned ON.

Memory Subsystems

The memory hierarchy of Ivy Bridge is as follows:

- L1 instruction cache: 32 KB, private to each core
- L1 data cache: 32 KB, private to each core
- L2 cache: 256 KB, private to each core
- L3 cache: 25 MB, shared by 10 cores in each socket
- Memory: 32 GB per socket, total of 64 GB per node

There are four 1866-MHz memory channels per socket. Each channel can be connected with up to two memory DIMMs. Of the eight memory DIMM slots for each socket, four are populated with 8-GB Error Correcting Code (ECC) registered DDR3 memory, for a total of 32 GB per socket. With two sockets in a node, the total memory per node is 64 GB.

Connecting the two sockets are two Intel QPI links running at a speed of 8.0 Giga-transfers (GT) per second. Each link contains separate lanes for the two directions. The total bandwidth (2 links x 2 directions) is 32 GB/sec.

Network Subsystem

The Ivy Bridge nodes are connected to the two fabrics (ib0 and ib1) of the Pleiades InfiniBand (IB) network via a dual-port, four-lane Fourteen Data Rate (4X FDR) IB Mezzanine card on each node, as well as the Mellanox FDR IB switches in the ICE X IB Premium Blade. The FDR runs at 14 Gb/sec per lane. With four lanes, the total bandwidth is 56 Gb/sec or about 7 GB/sec.

On each node, the IB Mezzanine card sits on a sister board next to the mother board, which contains the two processor sockets.

There are 18 nodes per Individual Rack Unit. These 18 nodes are connected to two Mellanox FDR IB switches in an ICE X IB Premium Blade to join the ib0 fabric. Another set of connections between the 18 nodes and a second Premium Blade is established for ib1.

Sandy Bridge Processors

sandybridge_processor_numbering.jpg

Core Labeling

The core labeling in Sandy Bridge is contiguous. That is, cores 0-7 are in the first socket and cores 8-15 are in the second socket.

When using the HPE MPT library, the environment variable **`MPI_DSM_DISTRIBUTE`** is set to ON by default for the Sandy Bridge nodes.

Instruction Set

A Sandy Bridge processor's execution hardware contains the Advanced Vector Extensions (AVX), a set of instructions for doing Single Instruction Multiple Data (SIMD) operations on Intel architecture processors. These extensions widen the vector registers from 128 bits to 256 bits, so the floating-point hardware can sustain 16 single-precision and 8 double-precision floating point operations per cycle. As a result, even though the CPU clock speed of the Sandy Bridge processor (2.6 GHz) is lower than that of older processor types on Pleiades, the floating-point performance can be higher for some applications.

AVX is supported in Intel compilers starting with version 11.1. However, Intel versions 12 and 13 compilers provide more optimizations for AVX and are recommended over version 11.1.

AVX is also supported in the GNU Compiler Collection starting with version 4.6. An application that is compiled with **`-xAVX`** can run on Sandy Bridge or Ivy Bridge.

TIP: If you want to have a single executable that will run on any of the Pleiades processor types, with suitable optimization to be determined at run time, you can compile your application with: **`-O3 -ipo -axCORE-AVX2 -xSSE4.2`**.

Hyperthreading

Hyperthreading is turned ON.

Turbo Boost

Turbo Boost is turned ON.

Memory Subsystems

The memory hierarchy of Sandy Bridge is as follows:

- L1 instruction cache: 32 KB, private to each core
- L1 data cache: 32 KB, private to each core
- L2 cache: 256 KB, private to each core
- L3 cache: 20 MB, shared by 8 cores in each socket
- Memory: 16 GB per socket, total of 32 GB per node

There are four 1600-MHz memory channels per socket. Each channel can be connected with up to two memory DIMMs. Of the eight memory DIMM slots for each socket, four are populated with 4-GB Error Correcting Code (ECC) registered DDR3 memory, for a total of 16 GB per socket. With two sockets in a node, the total memory per node is 32 GB. If there is a user requirement, some nodes could be configured with larger amounts of memory.

Connecting the two sockets are two Intel QPI links running at a speed of 8.0 Giga-transfers (GT) per second. Each link contains separate lanes for the two directions. The total bandwidth (2 links x 2 directions) is 32 GB/sec.

Network Subsystem

The Sandy Bridge nodes are connected to the two fabrics (ib0 and ib1) of the Pleiades InfiniBand (IB) network via the dual-port, four-lane Fourteen Data Rate (4x FDR) IB Mezzanine card on each node, as well as the Mellanox FDR IB switches in the ICE X IB Premium Blade. The FDR runs at 14 Gb/sec per lane. With four lanes, the total bandwidth is 56 Gb/sec or about 7 GB/sec.

On each node, the IB Mezzanine card sits on a sister board next to the mother board, which contains the two processor sockets.

There are 18 nodes per Individual Rack Unit. These 18 nodes are connected to two Mellanox FDR IB switches in an ICE X IB Premium Blade to join the ib0 fabric. Another set of connection between the 18 nodes and a second Premium Blade is established for ib1.

Hyperthreading

Hyperthreading is available and enabled on the Pleiades, Aitken, and Electra compute nodes. With hyperthreading, each physical core can function as two logical processors. This means that the operating system can assign two threads per core by assigning one thread to each logical processor.

Note: Hyperthreading is currently off on Aitken Rome nodes.

The following table shows the number of physical cores and potential logical processors available for each processor type:

Processor Model	Physical Cores (N)	Logical Processors (2N)
Sandy Bridge	16	32
Ivy Bridge	20	40
Haswell	24	48
Broadwell	28	56
Skylake	40	80
Cascade Lake	40	80

If you use hyperthreading, you can run an MPI code using 2*N* processes per node instead of *N* process per nodeâ   so you can use half the number of nodes for your job. Each process will be assigned to run on one logical processor; in reality, two processes are running on the same physical core.

Running two processes per core can take less than twice the wall-clock time compared to running only one process per coreâ   if one process does not keep the functional units in the core busy, and can share the resources in the core with another process.

Benefits and Drawbacks

Using hyperthreading can improve the overall throughput of your jobs, potentially saving standard billing unit (SBU) charges. Also, requesting half the usual number of nodes may allow your job to start running soonerâ   an added benefit when the systems are loaded with many jobs. However, using hyperthreading may not always result in better performance.

WARNING: Hyperthreading does not benefit all applications. Also, some applications may show improvement with some process counts but not with others, and there may be other unforeseen issues. Therefore, before using this technology in your production run, you should test your applications with and without hyperthreading. If your application runs more than two times slower with hyperthreading than without, do not use it.

Using Hyperthreading

Hyperthreading can improve the overall throughput, as demonstrated in the following example.

Example

Consider the following scenario with a job that uses 40 MPI ranks on Ivy Bridge. Without hyperthreading, we would specify:

```
#PBS -lselect=2:ncpus=20:mpiprocs=20:model=ivy
```

and the job will use 2 nodes with 20 processes per node. Suppose that the job takes 1000 seconds when run this way. If we run the job with hyperthreading, e.g.:

```
#PBS -lselect=1:ncpus=20:mpiprocs=40:model=ivy
```

then the job will use 1 node with all 40 processes running on that node. Suppose this job takes 1800 seconds to complete.

Without hyperthreading, we used 2 nodes for 1000 seconds (a total of 2000 node-seconds); with hyperthreading, we used 1 node for 1800 seconds (1800 node-seconds). Thus, under these circumstances, if you were interested in getting the best wall-clock time performance for a single job, you would use two nodes without hyperthreading. However, if you were interested in minimizing resource usage, especially with multiple jobs running simultaneously, using hyperthreading would save you 10% in SBU charges.

Mapping of Physical Core IDs and Logical Processor IDs

Mapping between the physical core IDs and the logical processor IDs is summarized in the following table. The value of *N* is 16, 20, 24, 28, and 40 for Sandy Bridge, Ivy Bridge, Haswell, Broadwell, and Skylake/Cascade Lake processor types, respectively.

Physical ID	Physical Core ID	Logical Processor ID
0	0	0 ; *N*
0	1	1 ; *N*+1
...	...	....
0	*N*/2 - 1	*N*/2 - 1; *N* + *N*/2 - 1
1	*N*/2	*N*/2 ; *N* + *N*/2
1	*N*/2 + 1	*N*/2 + 1; *N* + *N*/2 + 1
1	...	...
1	*N* - 1	*N* - 1; 2*N* - 1

Note: For additional mapping details, see the configuration diagrams at the for each processor type, or run the **`cat /proc/cpuinfo`** command on the specific node type.

Endeavour

Endeavour Configuration Details

Endeavour3 and Endeavour4 are two HPE Superdome Flex servers. They replace Endeavour1 and Endeavour2 as the NAS resources for supporting applications that need access to large cache-coherent, global shared-memory capabilities in a single system image (SSI).

Each of these two systems has 896 cores, six terabytes (TB) of memory, and a theoretical double-precision floating point peak performance of 77.414 teraflops. Each system is configured with a total of 32 sockets in this hierarchy:

- 28 cores and 192 gigabytes (GB) of memory in one socket.
- Four sockets in one chassis.
- Eight chassis in one system.

Processors and Memory Configuration in a Socket

At the processor level, Endeavour3 and 4 are based on the Intel Xeon Platinum 8280 processor, which is similar to Aitken's Intel Xeon Gold 6248 Cascade Lake processors.

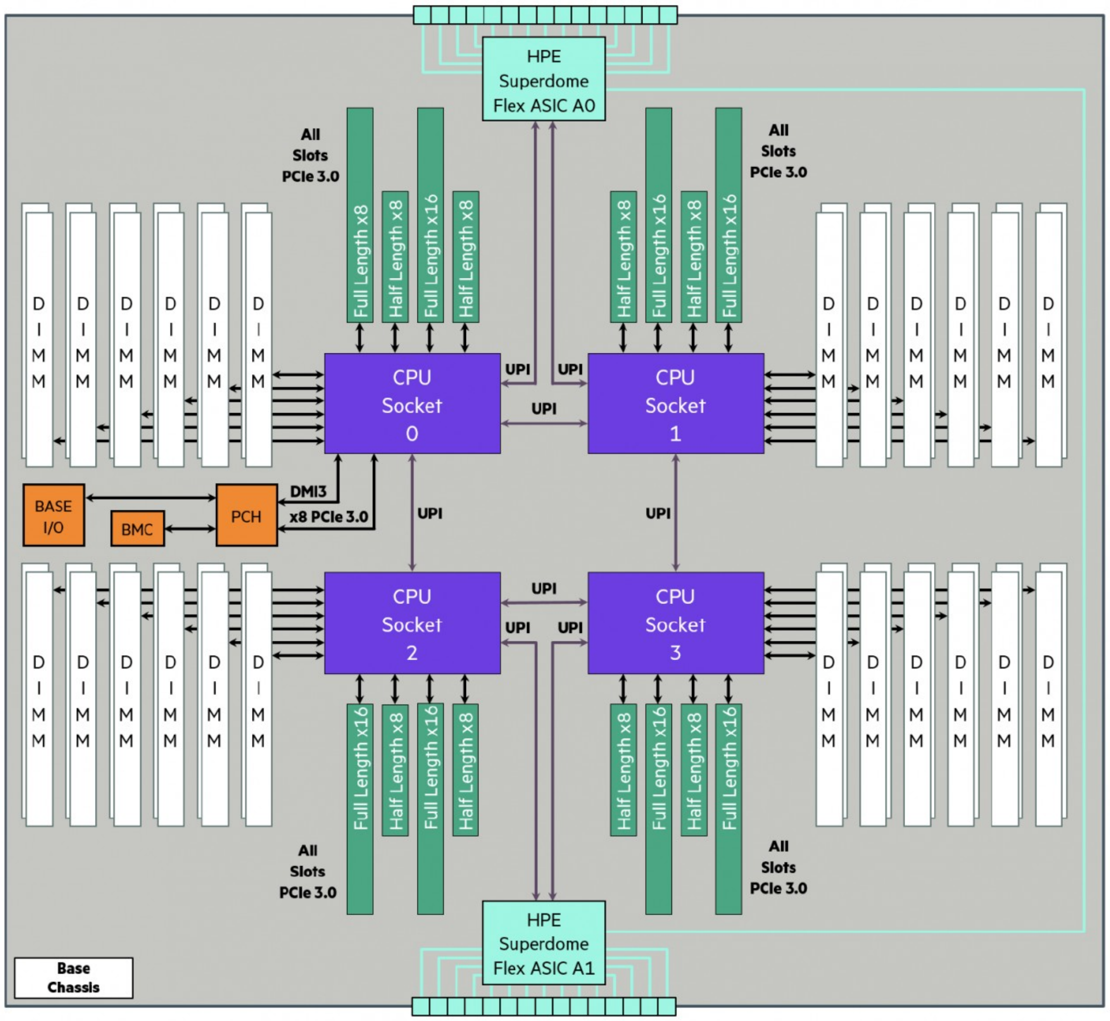
The main improvements provided by the Platinum 8280 processor are:

- The Endeavour systems have more cores per socket: 28 cores vs. 20 cores.
- The CPU clock speed is higher: 2.70 GHz vs. 2.50 GHz.
- The L3 cache size per socket is greater: 38.5 megabytes (MB) vs. 27.5 MB.
- The memory size per socket is increased: 192 GB vs. 96 GB.
- There is more DDR4 memory DIMM per memory channel: 32 GB vs. 16 GB.

For instruction sets, cache hierarchy, and memory subsystem information, see Preparing to Run on Endeavour.

Chassis Configuration

As shown in the following base chassis diagram, published by HPE, there are four sockets connected in a ring fashion via 10.4 GT/s Intel Ultra Path Interconnect (UPI) links. Each socket is also connected to one of two custom-designed HPE Superdome Flex ASICs, via an Intel UPI link, to send remote targeted cache-coherent data traffic to the external chassis.

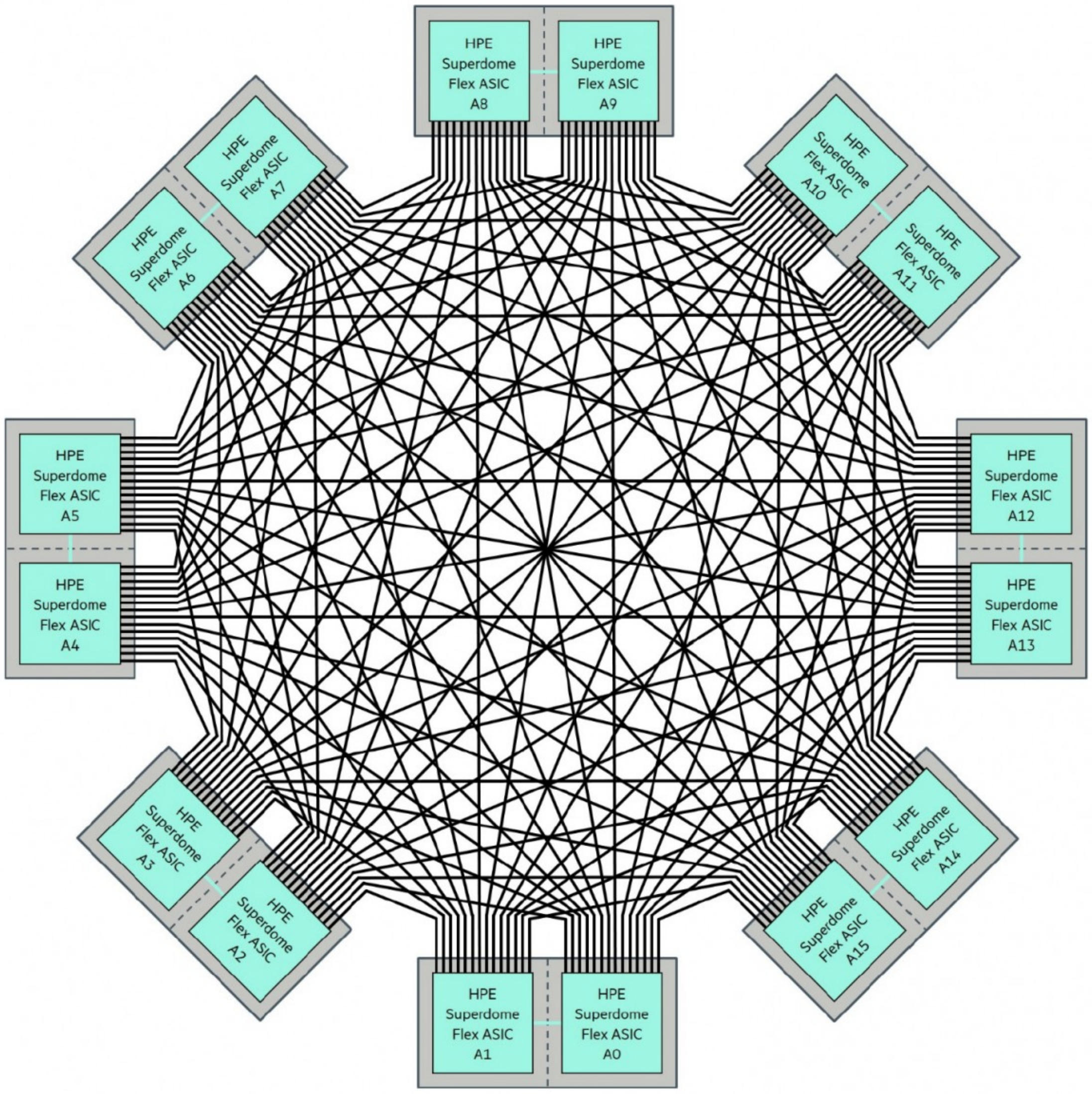


Memory latency within each socket is approximately 100 nanoseconds (ns). However, due to sub-NUMA design within a socket (see the Cascade Lake configuration diagram), it will be faster if the CPU that a process runs on is in the same sub-domain as the memory where the data is located. Memory latency between two sockets through a UPI link is approximately 130 ns.

All memory channels in each chassis are fully independent, and they can run simultaneously at DRAM data transfer rates up to 2,933 MegaTransfers per second (MT/s). This provides a theoretical total memory bandwidth of 564 GB/s; (memory bandwidth per socket is 141 GB/s). HPE's published STREAM TRIAD memory bandwidth is more than 360 GB/s per chassis.

System Interconnect

The HPE Superdome Flex server has a modular architecture based on a four-socket chassis. As shown in the following diagram published by HPE, the eight chassis of a 32-socket server are connected through a cabled crossbar interconnection fabric called *HPE Superdome Flex Grid*.


HPE Superdome Flex ASIC A8
HPE Superdome Flex ASIC A9
HPE Superdome Flex ASIC A7
HPE Superdome Flex ASIC A10
HPE Superdome Flex ASIC A6
HPE Superdome Flex ASIC A11
HPE Superdome Flex ASIC A5
HPE Superdome Flex ASIC A12
HPE Superdome Flex ASIC A4
HPE Superdome Flex ASIC A13
HPE Superdome Flex ASIC A3
HPE Superdome Flex ASIC A14
HPE Superdome Flex ASIC A2
HPE Superdome Flex ASIC A15
HPE Superdome Flex ASIC A1
HPE Superdome Flex ASIC A0

Each of the 16 application-specific integrated circuits (ASICs) in the grid has a very large directory cache to track the cache line state and maintain coherency across all attached processor sockets. A Superdome Flex ASIC (also known as NUMAlink 8) provides 16 *flex grid* ports, each capable of 13.3 GB/s data rates for maximum flex grid bandwidth. The total bi-sectional crossbar grid bandwidth for a 32-socket Superdome Flex server is more than 850 GB/s. With this crossbar interconnection, memory latency is minimized (< 400 ns), since there is only one hop between any two ASICs.

The HPE Superdome Flex Grid provides adaptive routing features designed specifically to enhance performance by routing traffic through the optimal latency path available, and provides superior uptime by automatically routing traffic around failed components.

Processor, Memory, and Network Subsystems Configuration

Detailed configuration statistics for Endeavour 3 and Endeavour 4 are listed below:

Processor, Memory, and Network Subsystems Statistics

Architecture	HPE Superdome Flex
CPU	Cascade Lake 28-Core Xeon 8280
CPU Base Clock	2.70 GHz
Hyperthreading	OFF
Turbo Boost	On (set to performance instead of powersave)
Maximum Double-Precision Flops per Cycle per Core	32
# of Cores/Socket	28
# of Sockets/Chassis	4
# of Chassis/System	8
Total # of cores/System	896
Total Double Precision TFlops/System	77.414
Memory/Core	6.85 GB
Memory/Socket	192 GB
Memory/System	6 TB
Memory Speed	2,933 MHz
System Interconnect	Superdome Flex ASICs (NUMAlink 8)
Topology	Crossbar

References

HPE Superdome Flex Server Architecture and RAS (PDF)

Preparing to Run on Endeavour

The Endeavour supercomputer comprises two nodes, Endeavour3 and Endeavour4. Each of the two systems includes 896 cores and six terabytes (TB) of global shared memory. See Endeavour Configuration Details for more hardware information.

The nodes use Pleiades front ends (PFEs) and filesystems, and share the Pleiades InfiniBand fabric. They use PBS server pbspl4.

Operating System

As of Dec. 20, 2021, most NAS systems are running the Red Hat Enterprise Linux-based Tri-Lab Operating System Stack (TOSS 3), including all of the PFEs.

However, both of the Endeavour nodes are still running SUSE Linux Enterprise Server 12 (SLES 12). Endeavour will be migrated to TOSS 3 in the near future. In the meantime, to build an executable to run on Endeavour, use an interactive PBS session connecting to Endeavour.

Connecting to Endeavour

To access the systems, submit PBS jobs from the Pleiades PFEs with the server name **`pbspl4`** included in the **`qsub`** command line or PBS script. You cannot log directly into Endeavour3 and 4 from the PFEs unless you have a PBS job running there.

Accessing Your Data

Use your Pleiades home filesystem (/u/*username*) and your Lustre filesystem (/nobackup) for both Pleiades and Endeavour.

For more information on using Lustre filesystems, see Lustre Basics and Lustre Best Practices.

Compiling and Running Your Code

There are no default modules for compilers, Message Passing Interface (MPI) components, or math libraries. To use these modules, you must first load them.

OpenMP Applications

Load an Intel compiler module and use the Intel compiler **`-qopenmp`** option to build the executable. Since the Cascade Lake processors used in Endeavour3 and 4 support AVX512, you can experiment with different compiler versions and flags for better performance. For example:

```
% module load comp-intel/2018.3.222
% ifort -O2 -qopenmp program.f
```

or

```
% module load comp-intel/2020.4.304
% ifort -Ofast -ipo -axCORE-AVX512,CORE-AVX2,AVX -xSSE4.2 -qopenmp program.f
```

Set a specific number of OpenMP threads in your PBS job, and use a thread-pinning method to ensure that the threads do not get in the way of each other on the same core or migrate from one core to another. For example:

```
#PBS ...

module load comp-intel/2020.4.304
setenv OMP_NUM_THREADS 28

/u/scicon/tools/bin/mbind.x -cs -t28 -v ./a.out

or

setenv KMP_AFFINITY compact # or: setenv KMP_AFFINITY scatter
./a.out > output

or

setenv KMP_AFFINITY disabled
dplace -x2 ./a.out > output
```

See Process/Thread Pinning Overview for more information about different pinning methods.

Note: The default OMP_STACKSIZE is set to 200 megabytes (MB). If your application experiences segmentation fault (segfault), experiment with increasing the $OMP_STACKSIZE value to see if it helps.

Applications that Require Scientific and Math Libraries

The Intel Math Kernel Library (MKL) is included in Intel compiler modules (versions 11.1 and later). See MKL to learn how to link to MKL libraries. In many cases, the following commands may be sufficient:

```
% module load comp-intel/2020.4.304
% ifort -O2 program.f -mkl
```

Note: By itself, the `-mkl` option implies `-mkl=parallel`, which will link to the threaded MKL library. If your application can benefit from using multiple OpenMP threads when the MKL routines are called, you should also set the environment variable `OMP_NUM_THREADS`, as shown in the OpenMP example in the previous section.

If you want to use a non-threaded MKL library, change the `-mkl` option to `-mkl=sequential`.

MPI Applications

HPE's latest Message Passing Toolkit (MPT) library is recommended. Load the library as shown in the following example:

```
% module load comp-intel/2020.4.304
% module load mpi-hpe/mpt
% ifort -O2 program.f -lmpi
```

During runtime, load the Intel compiler and the MPT modules and use `mpiexec` to launch the executable:

```
#PBS ...
module load comp-intel/2020.4.304
module load mpi-hpe/mpt

mpiexec -np xx ./a.out > output
```

Running PBS Jobs

Although the Endeavour3 and Endeavour4 nodes use PFEs and Pleiades filesystems, and share the Pleiades InfiniBand fabric for I/O, they use a separate PBS server: pbspl4. Therefore, PBS jobs cannot run across the Endeavour3 and 4 and Pleiades compute nodes.

Each of the Endeavour nodes has 32 sockets, and each socket includes 28 cores and 192 gigabytes (GB) of physical memory. In each socket, PBS has access to 28 cores and 185 GB of memory, which is known as a minimum allocatable unit (MAU). In addition, one of the sockets is set aside for system operations, leaving 31 sockets available for user jobs.

PBS does not allow jobs to share resources in the same socket. However, the amount of resources in a socket that a job will receive is restricted to the amount requested. This restriction is enforced by the Linux kernel **`cgroups`** feature. For example, if you request one core and 100 GB of memory, one socket will be assigned exclusively for your job, but 27 cores and 85 GB of memory of the socket will not be accessible by your job.

When job accounting is enabled, your job will be charged by the number of MAUs assigned exclusively for your job.

The maximum amount of resources a job can request is:

- **Endeavour3:** 868 cores with 5,750 GB are the maximum resources available for PBS jobs.
- **Endeavour4:** 868 cores with 5,750 GB are the maximum resources available for PBS jobs.

Endeavour PBS Queues

The PBS server pbspl4 manages jobs for Endeavour3 and 4 (**`:model=cas_end`**) and the NAS V100 GPU nodes (**`:model=sky_gpu`** and **`:model=cas_gpu`**). Among the queues listed by the command **`qstat -Q @pbspl4`**, you can run jobs on the two systems using the e_normal, e_long, e_vlong, and e_debug queues. You can also find detailed settings of each queue, such as the maximum walltime and the default or minimum ncpus and memory, using the following command line (the e_normal queue is used in the example):

```
pfe% qstat -fQ e_normal@pbspl4
```

PBS Commands

Run PBS commands (such as **`qsub`**, **`qstat`**, and **`qdel`**, etc.) from the PFEs and specify the PBS server, pbspl4. For example:

```
pfe21% qsub -q queue_name@pbspl4 job_script
pfe21% qstat -nu username @pbspl4
pfe21% qstat job_id.pbspl4
pfe21% qdel job_id.pbspl4
```

If you do not normally run jobs on Pleiades compute nodes, and your primary workload is on Endeavour3 and Endeavour4, you may want to consider setting the **`PBS_DEFAULT`** environment variable to pbspl4. This allows you to simplify the PBS commands, as follows:

```
pfe21% setenv PBS_DEFAULT pbspl4
pfe21% qsub job_script
pfe21% qstat -nu username
```

```
pfe21% qstat job_id
pfe21% qdel job_id
```

Job Submission Examples

The following examples demonstrate how PBS allocates resources for your job.

Note: If you do not specify a resource attribute (such as ncpus or mem), a default value for that attribute will be assigned to your job.

- To request using all the resources in two MAUs:

  ```
  pfe21% qsub -I -lselect=2:ncpus=28:mem=185G:model=cas_end -q queue_name@pbspl4
  ```
- To request 56 cores:

  ```
  pfe21% qsub -I -lselect=1:ncpus=56:model=cas_end -q queue_name@pbspl4
  ```

  Your job will be given access to 56 cores across in two sockets and only the default amount of memory (32G).
- To request 1,850 GB of memory:

  ```
  pfe21% qsub -I -lselect=1:mem=1850G:model=cas_end -q queue_name@pbspl4
  ```

  Your job will be given access to 1,850 GB of memory spread in ten sockets and only the default number of cores (one).
- If you do not specify which Endeavour node to run your job, PBS will assign one for you. The following example specifies Endeavour3 and requests 28 cores and 185 GB of memory:

  ```
  pfe21% qsub -I -lselect=host=endeavour3:ncpus=28:mem=185G:model=cas_end -q queue_name@pbspl4
  ```

Job Accounting

The Standard Billing Unit (SBU) rate for using one MAU per hour is 1.31.

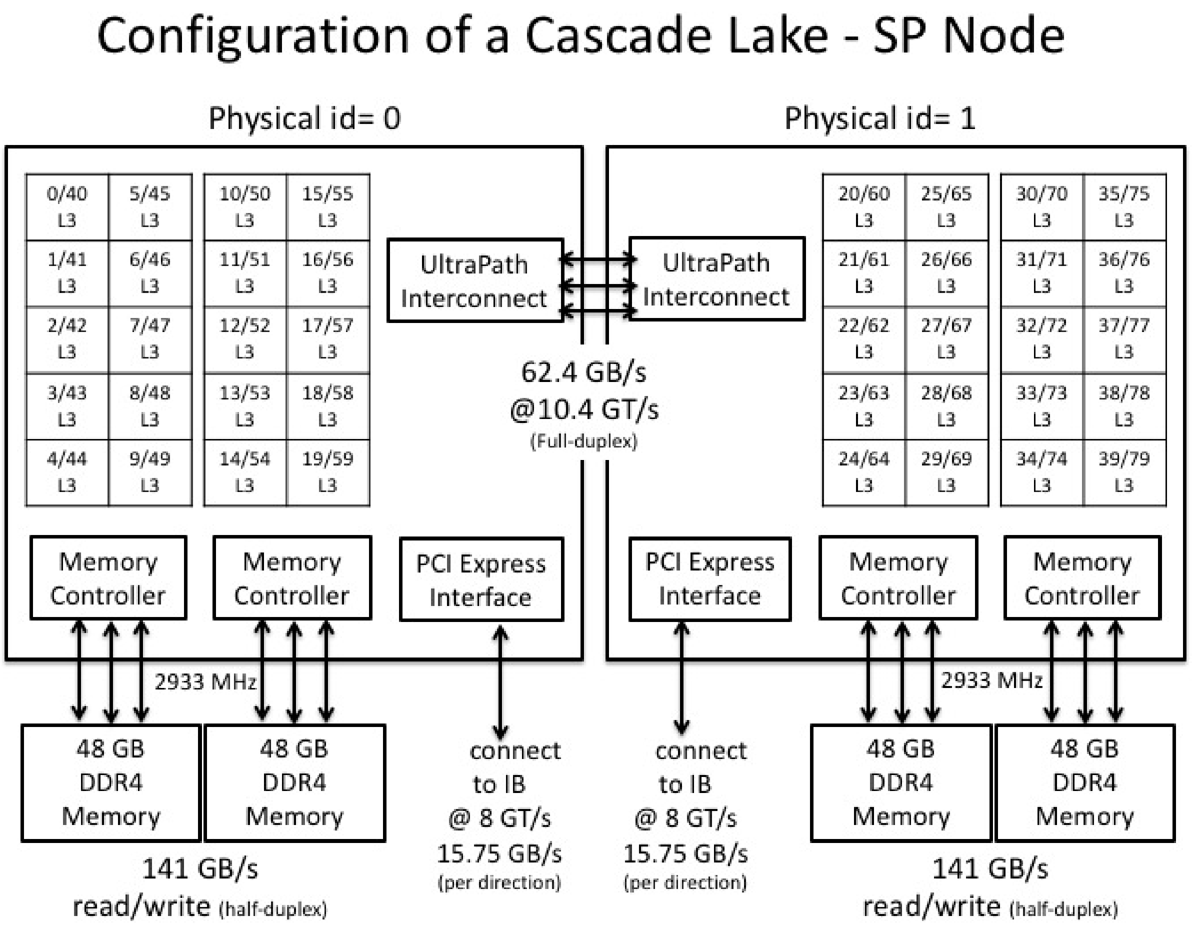
As of October 1, 2019, your group's HECC SBU allocation is charged for usage on any HECC resources including Pleiades, Electra, Aitken, and Endeavour. The output from the **acct_ytd** command does not have separate entries for the different systems. To find Endeavour3 and Endeavour4 usage for your group, use the **acct_query** command instead.

For example, to view the statistics of all of your jobs that ran on Endeavour3 on March 10, 2021, type:

```
pfe21% acct_query -c endeavour3 -d 03/10/21 -u your_username -olow
```

See Job Accounting Utilities to learn more about using the **acct_ytd** and **acct_query** tools.

Cascade Lake Processors



Notes:

- Each small square in the diagram represents a combination of a physical core and a L3 cache slice. For each physical core, there are two logical core labelings obtained from the `"processor"` entry of the `cat /proc/cpuinfo` output.
- Although Cascade Lake processors support up to three Intel Ultra Path Interconnect (UPI) links with a bandwidth of 62.4 GB/s, the HPE 8600 Saxon board used for the Cascade Lake nodes at NAS implements only two links, with a bandwidth of 41.6 GB/s.

The Intel Cascade Lake processor incorporated into the Aitken cluster is the 20-core Xeon Gold 6248 model. Its base clock speed is 2.5 GHz for non-AVX, 1.9 GHz for AVX2, and 1.6 GHz for AVX-512. It uses an enhanced 14-nanometer (nm) fabrication process and the Skylake microarchitecture with some optimization.

Each Aitken Cascade Lake node contains two Cascade Lake processors and uses a dual single-port 100 Gbits/s Enhanced Data Rate (EDR) Mezzanine card to connect outward (see Inter-node Network). The use of four 200 Gbits/s High Data Rate (HDR) switches per enclosure forms the MPI and I/O InfiniBand fabrics within the Aitken cluster. Connecting the Aitken cluster to the Pleiades filesystems relies on additional sets of HDR switches and cables.

Instruction Sets

In addition to the instruction sets SSE, SSE2, SSE3, Supplemental SSE3, SSE4.1, SSE4.2, AVX, AVX2, AVX-512, and AVX512[F,CD,BW,DQ,VL], which are available in its Skylake predecessor, Cascade Lake also includes the new AVX-512 Vector Neural Network Instructions (VNNI), which provide significant, more efficient deep-learning inference acceleration. Cascade Lake also introduces in-hardware mitigations for the Spectre and Meltdown security flaws.

With 512-bit floating-point vector registers and two floating-point functional units, each capable of Fused Multiply-Add (FMA), a Cascade Lake core can deliver 32 double-precision floating-point operations per cycle.

Use the Intel compiler flag **`-xCORE-AVX512`** for Skylake and Cascade Lake-SP specific optimizations. The optimization flag **`-qopt-zmm-usage=high -xCORE-AVX512`** may benefit floating-point heavy applications running on Skylake and Cascade Lake.

Tip: If you want a single executable that will run on any of the Aitken, Electra, and Pleiades processor types, with suitable optimization to be determined at run time, you can compile your application using the option:
`-O3 -ipo -axCORE-AVX512,CORE-AVX2 -xAVX`.

Hyperthreading

Hyperthreading is turned ON.

Turbo Boost

Turbo Boost is turned ON. Maximum Turbo Frequency is 3.90 GHz for non-AVX, and 3.8 GHz for AVX2 and AVX-512.

Cache Hierarchy

The cache hierarchy of Cascade Lake is as follows:

- L1 instruction cache: 32 KB, private to each core; 64 sets; 64 B/line; 8-way
- L1 data cache: 32 KB, private to each core; 64 sets; 64 B/line; 8-way; fastest latency: 4 cycles; 128 B/cycle load bandwidth; 64 B/cycle store bandwidth; write-back policy
- L2 cache: 1 MB, private to each core; 64 B/line; 16-way; fastest latency: 14 cycles; 64 B/cycle bandwidth to L1 cache; write-back policy
- L3 cache: shared non-inclusive 1.375 MB/core; total of 27.5 MB, shared by 20 cores in each socket; 2,048 sets; 64 B/line; fully associative; fastest latency: 50 - 70 cycles; write-back policy

Memory Subsystem

Like Skylake, there are two sub-NUMA clusters in each Cascade Lake socket, creating two localization domains. There are three memory channels per sub-NUMA cluster. Each channel can be connected with up to two memory DIMMs. For the Aitken Cascade Lake configuration, there is one 16-gigabyte (GB) dual rank DDR4 DIMM with error correcting code (ECC) support per channel. In total, the amount of memory is 48 GB per sub-NUMA cluster, 96 GB per socket, and 192 GB per node.

The speed of each memory channel is increased from 2,666 MHz in Skylake to 2,933 MHz in Cascade Lake. An 8-byte read or write can take place per cycle per channel. With a total of six memory channels, the total half-duplex memory bandwidth is approximately 141 GB/s per socket.

On-chip Interconnect

The on-chip architecture of Cascade Lake SP uses the same mesh layout as that of Skylakeâ where the cores and L3 caches are organized in rows and columnsâ instead of the ring architecture used by earlier Xeon processors.

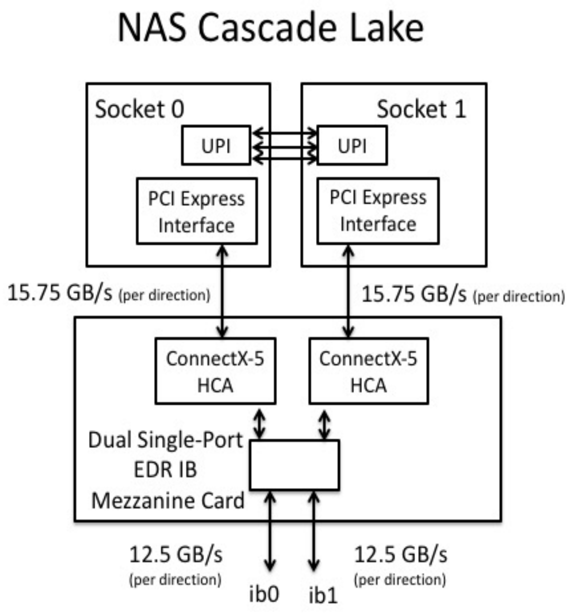
Inter-socket Interconnect

The Cascade Lake processors support up to three UPI links. The configuration at NAS has two links enabled. The UPI runs at a speed of 10.4 gigatransfers per second (GT/s). Each link contains separate lanes for the two directions. The total full-duplex bandwidth is 62.4 GB/s with three links and 41.6 GB/s with two links.

Inter-node Network

Like the network subsystems of the Pleiades Haswell and Broadwell nodes, and the Electra Broadwell and Skylake nodes, each Cascade Lake node uses two PCI Express interfaces (one from each socket) to connect to the Aitken and Pleiades InfiniBand (IB) fabrics. The use of PCIe 3.0 16-lane (x16) links enables a maximum bandwidth of 15.75 GB/s for each direction.

Like the Electra Skylake nodes, the Aitken Cascade Lake inter-node network makes use of the 100 Gbits/s EDR technology. As shown below, one PCIe is connected to the ib0 fabric via a single-port, four-lane (4X) EDR host channel adapter (HCA), in a dual single-port EDR IB mezzanine card, with an effective bandwidth of 100 Gbits/s (that is, 12.5 GB/s) for each direction. The other PCIe is connected to the ib1 fabric via another single-port 4x EDR HCA in the same mezzanine card.


NAS Cascade Lake
Socket 0
Socket 1
UPI
UPI
PCI Express Interface
PCI Express Interface
15.75 GB/s (per direction)
15.75 GB/s (per direction)
ConnectX-5 HCA
ConnectX-5 HCA
Dual Single-Port EDR IB Mezzanine Card
12.5 GB/s (per direction)
12.5 GB/s (per direction)
ib0
ib1

There are 1,152 Cascade Lake nodes in the Aitken cluster, which uses the liquid-cooled HPE 8600 system architecture. The nodes are partitioned as follows:

- Four E-cells (288 nodes per E-cell)
- Two compute racks and one cooling rack per E-cell (144 nodes per compute rack)
- Four enclosures (IRUs) per compute rack (36 nodes per enclosure)
- Nine compute trays (HPE XA730i "Saxon" blades) per enclosure (four nodes per tray)

The connection of the compute nodes relies on the use of four standard IB HDR switch blades per enclosure. Two of these switch blades facilitate the ib0 fabric, while the other two facilitate the ib1 fabric. Each switch blade has one 40-port ConnectX-6 ASIC with 18 ports connecting to the compute nodes, and 22 ports available for switch-to-switch links and connections to external targets. The connection of the 1,152 Cascade Lake nodes forms a 6-dimensional enhanced hypercube.

Notes:

- There are eight ASICs per enclosure in Electra Skylake, while there are four in Aitken Cascade Lake. This reduces a 1,152-node topology from seven dimensions in Electra Skylake to six dimensions in Aitken Cascade Lake.
- Only two out of the three UPI links are enabled.

References

- Intel Xeon Gold 6248 Processor
- Second Generation Intel Xeon Scalable Processors
- Intel 64 and IA-32 Architectures Optimization Reference Manual (Apr 2019 edition)
- HPE SGI 8600 QuickSpecs (Apr 15, 2019 edition)

GPU Nodes

Changes to PBS Job Requests for V100 GPU Resources

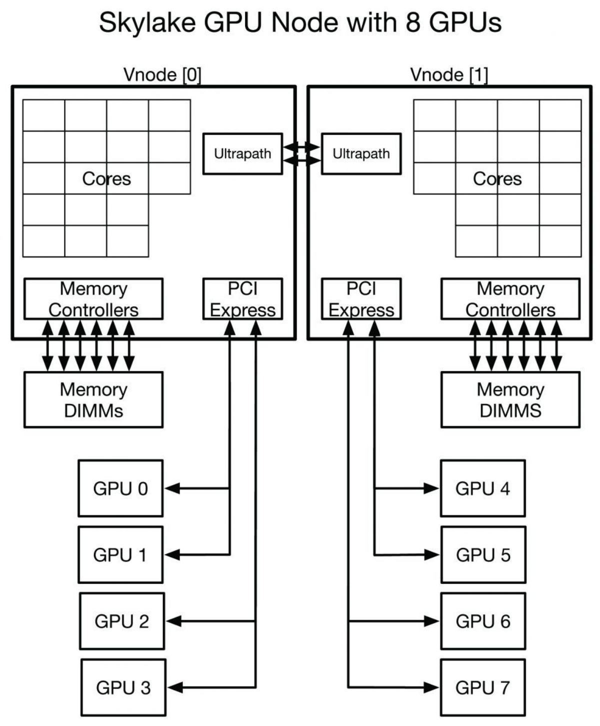
As of July 16, 2020, PBS handles NVIDIA Volta V100 GPU nodes differently than in the past. The new configuration allows multiple PBS jobs to share a single node. This means that you can request and access a subset of GPUs on the node for your job, so each job will utilize only the GPUs it needs, leaving the rest of the GPUs on the node available to other jobs. This enables more efficient use of the V100 nodes.

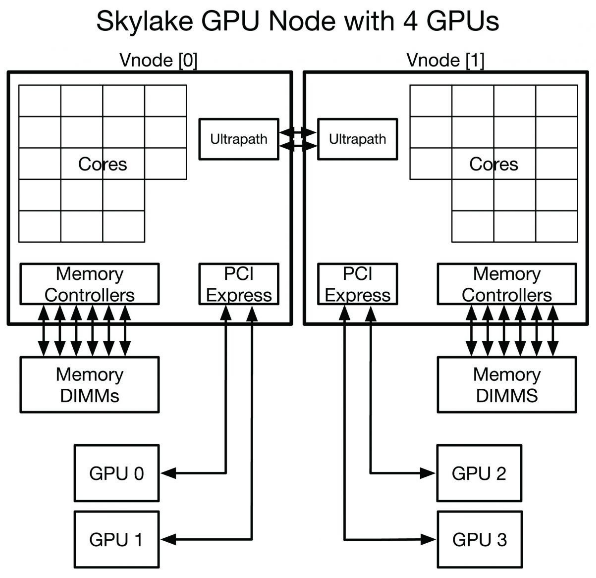
Instead of each node being treated as one unit for exclusive access by a single job, the nodes are logically split into two virtual nodes (*vnodes)*â   one for each socket and its associated CPU cores, GPUs, and memory. Unless specified otherwise, a job will have exclusive access only to the resources specified in the job request. Other jobs might run on the same vnodes at the same time but use different CPUs, GPUs and memory.

Note: The **`sky_gpu`** nodes are used below to explain the changes. The explanation also applies to the **`cas_gpu`** nodes except that there are 24 CPU cores per socket in **`cas_gpu`**.

Configuration of Skylake GPU Vnodes

The following diagrams show the logical splitting of four-GPU and eight-GPU Skylake nodes into vnodes. This configuration is similar to the standard Electra Skylake node, except that the GPU version has 18 cores and 192 gigabytes (GB) of memory per socket.





Requesting V100 GPU Resources

As a quick review, the following definitions are paraphrased from the Altair PBS Professional User's Guide:

- A host is any computer. Hosts where jobs run are often called nodes.
- A virtual node, or vnode, is an abstract object representing a set of resources that form a usable part of a machine. This could be an entire host, or a nodeboard or a blade. A single host can be made up of multiple vnodes.
- A chunk specifies the value of each resource in a set of resources that are to be allocated as a unit to a job. It is the smallest set of resources to be allocated to a job. All of a chunk is taken from a single host. One chunk may be broken across vnodes, but all participating vnodes must be from the same host.

When you request non-v100 node types using other PBS queues, the job request looks similar to the following:

```
#PBS -l select=100:model=ivy:ncpus=20
```

where **`100`** indicates how many "chunks" of resources are requested. Each chunk is assigned to a different node, and the job has exclusive access to all the resources on the node. This is not the case with the new **`v100`** queue.

Specifying Chunk Resources for the v100 Queue

The chunk resources that must be specified for the **`v100`** queue are:

ncpus
: The number of cores. Both hyperthreads on each core will be included in the chunk.

ngpus
: The number of GPUs. PBS sets the environment variable CUDA_VISIBLE_DEVICES to a list of the appropriate IDs to access the assigned GPUs.

mpiprocs
: The number of MPI ranks.

ompthreads
: PBS sets this into the OMP_NUM_THREADS environment variable.

mem
: The amount of CPU memory. PBS enforces this limit.

Specifying the Place Statement

In addition, you must specify the **`place`** statement. For our purposes, this statement has the form **`place=`*`arrangement:sharing.`***

Arrangement Attribute

Possible values for the ***`arrangement`*** attribute:

free
: Place job chunks on any vnodes.

pack
: All chunks will be taken from one host.

scatter
: Only one chunk is taken from any host.

vscatter

Only one chunk is taken from any vnode.

Sharing Attribute

Possible values for the ***sharing*** attribute:

excl
    Only this job uses the vnodes chosen.
exclhost
    The entire host is allocated to the job.
shared
    This job can share the vnodes chosen.

Resource Request Examples

For example, suppose in the past you wanted four v100 nodes all to yourself. You probably specified something like:

```
#PBS -l select=4:model=sky_gpu:naccelerators=4
```

The equivalent new request is:

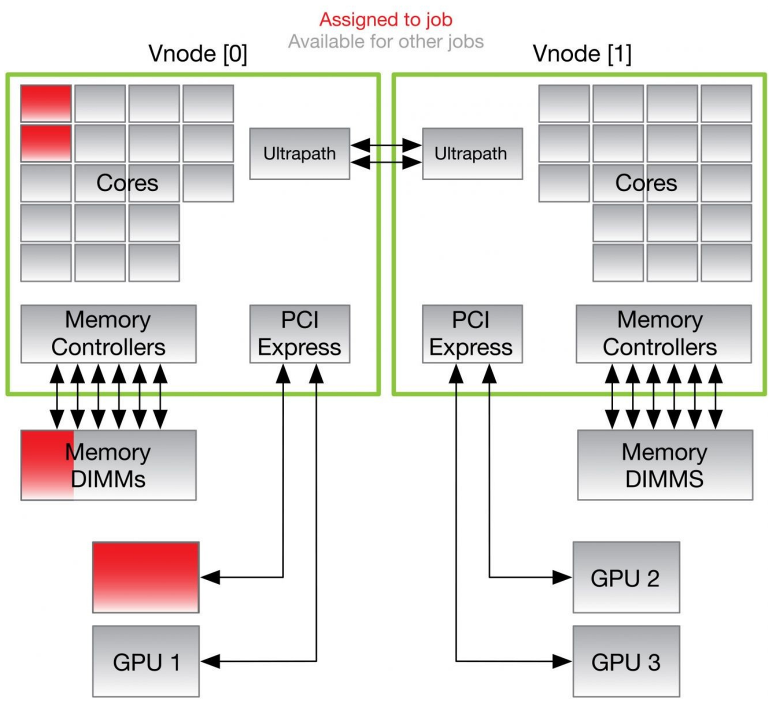
```
#PBS -l select=4:model=sky_gpu:mpiprocs=1:ncpus=36:ngpus=4:mem=300g
#PBS -l place=scatter:excl
```

The number of chunks (4) and the model type (**sky_gpu**) stay the same. However, in order to get access to all the cores, GPUs, and memory on the nodes, you must specify these resources explicitly. Furthermore, on the **place** statement, you must specify exclusive access (**excl**).

Now suppose you want to use just one GPU for a non-MPI program, where the program uses two OpenMP threads on two cores and 40 GB of memory. To make this request:

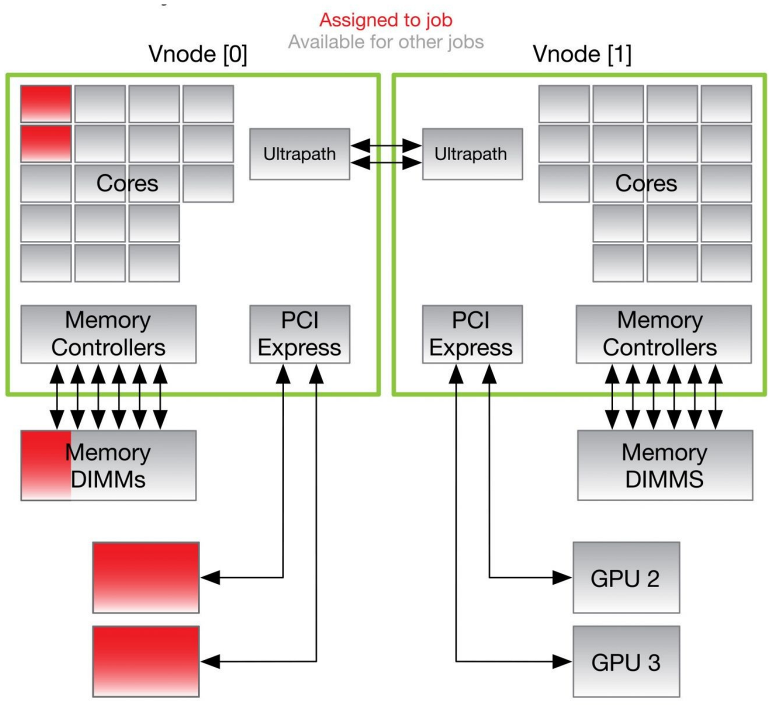
```
#PBS -l select=1:model=sky_gpu:ngpus=1:ncpus=2:ompthreads=2:mem=40g
#PBS -l place=vscatter:shared
```

Because the program is non-MPI, you don't need to specify **mpiprocs**. The other resource values come directly from your requirements. The resulting assignment is shown in the following diagram:



Note: The particular cores, GPUs, and memory might differ, but all resources will come from the same vnode. (Without **`vscatter`**, the resources might be split across both vnodes in a host.)

To give this program access to two GPUs, just change **`ngpus=1`** to **`ngpus=2`**. This results in the following assignment:

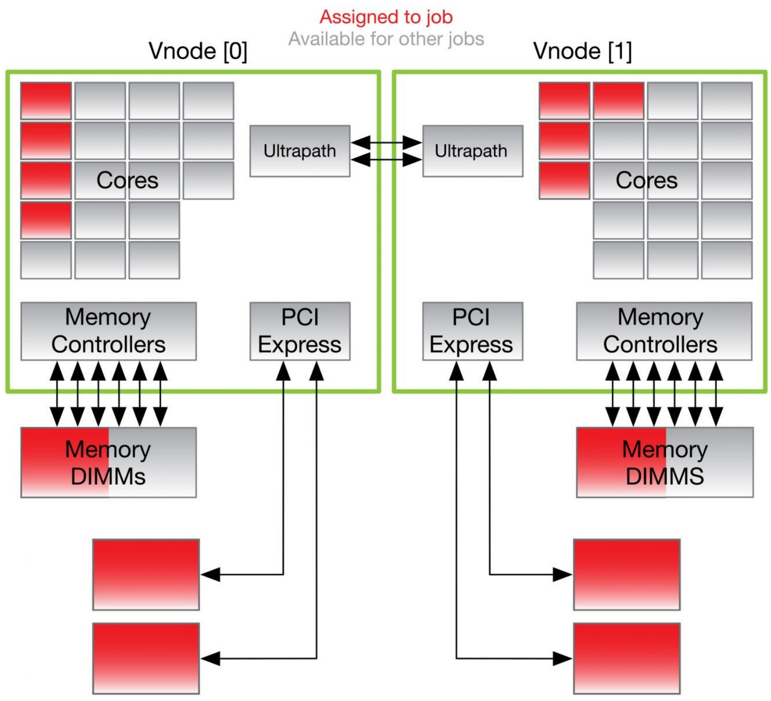


Because these resources comprise significantly less than a whole **`sky_gpu`** node, the **`place`** statement indicates that the job can share its node with other jobs.

Suppose, instead, you have an MPI program with four ranks where each rank uses one GPU, two CPU cores, and 50 GB of memory:

```
#PBS -l select=4:model=sky_gpu:ngpus=1:mpiprocs=1:ncpus=2:ompthreads=2:mem=50g
#PBS -l place=pack:shared
```

This results in the following assignment:

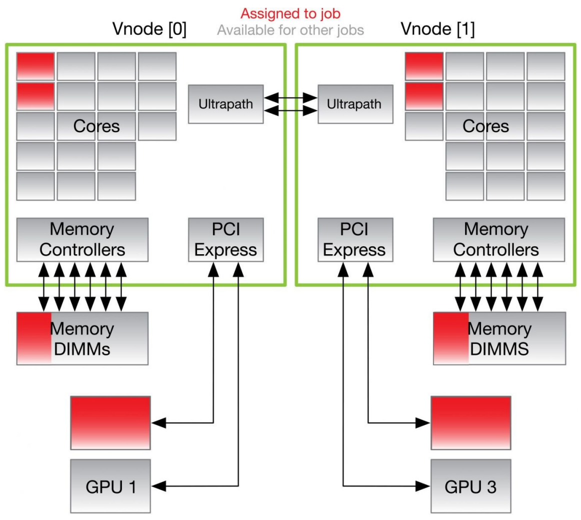


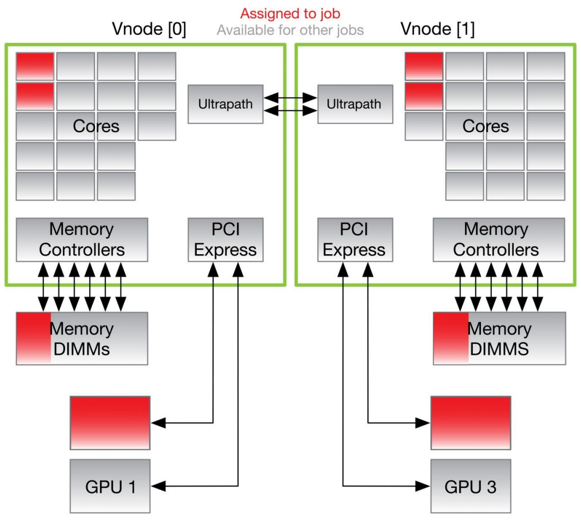
Here, the **place** value of **pack** means all the chunks will come from the same host, which usually gives the best MPI performance. On the other hand, if MPI is only a small part of the job, you might use **place=free:shared** so you don't need to wait for all resources to be available on a single hostâ   the resources can come from multiple hosts, wherever they are available.

You might want the GPUs to be on different sockets. In this case, change the **place** specification to:

```
#PBS -l place=vscatter:shared
```

Each of the four chunks will be placed on a different vnode, which means the chunks' GPUs will be attached to different sockets. The **sky_gpu** nodes have only two sockets, so the job will be spread over multiple hosts:





For more information, see Requesting GPU Resources.

If you have any questions, please contact the NAS Control Room at (800) 331-8737, (650) 604-4444, or support@nas.nasa.gov.

Using GPU Nodes

GPU Node Reservation: Thirty Cascade Lake-V100 GPU nodes (cas_gpu) are currently reserved for a special project. During this time, we recommend using the Skylake-V100 GPU nodes (sky_gpu) for your jobs requiring V100 GPUs.
A graphics processing unit (GPU) is a hardware device that can accelerate some computer codes and algorithms. This article describes the two types of GPU nodes currently available at NAS. For information about running your PBS jobs on the GPU nodes, see Requesting GPU Resources. For information about developing code for the GPU nodes, see GPU Programming.

For more general information on GPUs, see the GPGPU article on Wikipedia and the NVIDIA website, which has information for developers.

If you have an application that can take advantage of GPU technology, you can use one of the NAS GPU models:

- model=san_gpu

  64 nodes total. Each node contains two Sandy Bridge CPU sockets and one NVIDIA Kepler K40 GPU coprocessor (card) connected via PCI Express bus.

  The Sandy Bridge sockets used in the **san_gpu** nodes and those in the other Pleiades Sandy Bridge nodes are both Intel Xeon E5-2670. However, the size of memory in the sockets of the **san_gpu** nodes is double that of the other Pleiades Sandy Bridge nodes.
- model=sky_gpu

  19 nodes total. There are two different configurations:

  - 17* nodes, each containing two Skylake CPU sockets and **four** NVIDIA Volta V100 GPU cards, where the CPUs and GPUs are connected via PCI Express bus
  - Two nodes, each containing two Skylake CPU sockets and **eight** NVIDIA Volta V100 GPU cards, where the CPUs and GPUs are connected via PCI Express bus

  The Skylake sockets used in the **sky_gpu** nodes are Intel Xeon Gold 6154 while those in the Electra Skylake nodes are Intel Xeon Gold 6148. The main differences in these two configurations (**sky_gpu** vs. Electra Skylake) are: (1) number of cores - 18 vs. 20; (2) CPU clock speed - 3.0 vs. 2.4 GHz; (3) L3 cache size - 24.75 vs. 27.5 MB; (4) memory size - 384 vs. 192 GB; and (5) number of UPI links - 3 vs. 2.

  * Three of the 17 nodes are reserved for a special project; 14 nodes are available for use.
- model=cas_gpu

  38* nodes total. Each node contains two Cascade Lake CPU sockets and four NVIDIA Volta V100 GPU cards, where the CPUs and GPUs are connected via PCI Express bus.

  The Cascade Lake sockets used in the cas_gpu nodes are Intel Xeon Platinum 8268 while those in the Aitken Cascade Lake nodes are Intel Xeon Gold 6248. The main differences in these two configurations (cas_gpu vs Aitken Cascade Lake) are: (1) number of cores - 24 vs 20; (2) CPU clock speed - 2.9 vs. 2.5 GHz; (3) L3 cache size - 35.75 vs. 27.5 MB; and (4) memory size (384 vs 192 GB).

  * Four of the 38 nodes are reserved for a special project. 34 nodes are available for public use.

The SBU rates for the GPU nodes are:

- `san_gpu`: 0.94
- `sky_gpu` with four V100s: 9.82
- `sky_gpu` with eight V100s: 15.55
- `cas_gpu` with four V100s: 9.82

Node Details			
	san_gpu	**sky_gpu**	**cas_gpu**
Architecture	In-house	Apollo 6500 Gen10	Apollo 6500 Gen10
Total # of Nodes	64	19	38
Host name	r313i[0-3]n[0-15]	r101i0n[0-11,14-15], r101i1n[0-2] and r101i0n[12-13]*	r101i2n[0-17], r101i3n[0-15] and r101i4n[0-3]
CPU Host			
Processors Model	8-core Xeon E5-2670	18-core Xeon Gold 6154	24-core Xeon Platinum 8268
# of CPU Cores/Node	16	36	48
CPU-Clock	2.6 GHz	3.0 GHz	2.9 GHz
Maximum Double Precision Floating Point Operations per cycle per core	8	32	32
Total CPU Double Precision Flops/Node	332.8 GFlops	3,456 GFlops	4,454 GFlops
Memory/Node	64 GB (DDR3)	384 GB (DDR4)	384 GB (DDR4)
GPU			
Device Name	Tesla K40m	Tesla V100-SXM2-32GB	Tesla V100-SXM2-32GB
Clock Rate (Base/Boost)	745 MHz/875 MHz	1290 MHz/1530 MHz	1290 MHz/1530 MHz
# of coprocessors/Node	1	4 (17 nodes) or 8 (2 nodes)	4
# of Single Precision CUDA Cores/coprocessor	2880	5120	5120
# of Double Precision CUDA Cores/coprocessor	960	2560	2560
# of Tensor Cores/coprocessor	N/A	640	640
Total GPU Double Precision Flops/coprocessor (Base/Boost)	1.43 TFlops/1.68 TFlops	6.6 TFlops/7.8 TFlops	6.6 TFlops/7.8 TFlops
Total GPU Double Precision Flops/Node (Base/Boost)	1.43 TFlops/1.68 TFlops	26.4 TFlops/31.2 TFlops for 4 V100 and 52.8 TFlops/62.4 TFlops for 8 V100	26.4 TFlops/31.2 TFlops
L2 Cache	1536 KB	6144 KB	6144 KB
Global Memory/coprocessor	12 GB (GDDR5)	32 GB (HBM2)	32 GB (HBM2)
Memory Clock Rate	3004 MHz	877 MHz	877 MHz
Memory Bus Width	384 bits	4096 bits	4096 bits
Memory Bandwidth	288 GB/s	900 GB/s	900 GB/s

GPU-to-GPU Communication	N/A	SXM2 mezzanine connector with NVLink 2.0 with 300 GB/s bi-directional bandwidth	SXM2 mezzanine connector with NVLink 2.0 with 300 GB/s bi-directional bandwidth
PGI Compiler Option	-ta=tesla:cc35	-ta=tesla:cc70	-ta=tesla:cc70
Inter-node Network			
IB Device on Node	1 FDR 2-port card	2 EDR 100 Gb 2-port cards	2 EDR 100 Gb 2-port cards
Local Disk			
SSD/node	N/A	** 3.2 TB raw, 1.6 TB usable (mirrored) (2 x 1.6 TB SAS SSD connected via an internal smart Raid card)	** 3.2 TB raw, 1.6 TB usable (mirrored) (2 x 1.6 TB SAS SSD connected via an internal smart Raid card)

* r101i0n[12-13] are for the two nodes containing eight V100 cards.

** Not yet available for public usage, pending configuration decisions.

For more hardware details, access a GPU node through a PBS session and run one of the following commands:

```
For CPU info:

cat /proc/cpuinfo

For host memory info:

cat /proc/meminfo


For GPU info:

/usr/bin/nvidia-smi -q

or load a PGI compiler module, for example, comp-pgi/20.4, and run the command pgaccelinfo.


module use -a /nasa/modulefiles/testing
module avail comp-pgi
---- /nasa/modulefiles/testing ----
comp-pgi/17.10 comp-pgi/18.10 comp-pgi/18.4 comp-pgi/19.10 comp-pgi/19.5 comp-pgi/20.4
---- /nasa/modulefiles/sles12 -----
comp-pgi/16.10 comp-pgi/17.1

module load comp-pgi/20.4
pgaccelinfo
```

Currently, no direct communication exists between a GPU on one node and a GPU on another node. If such communication is required, the data must go from the GPU to the CPU via the PCI Express bus, and from one CPU to another via InfiniBand network using MPI.

GPU Programming

To develop code for use on the GPU nodes, you can choose one of three programming models: the directive-based OpenACC model, the language-based CUDA model, or the language-based OpenCL model. Each model is described in this article.

For specific information about NVIDIA GPUs and the programming models for them, see the NVIDIA Developer Zone.

The Directive-Based OpenACC Model

OpenACC is a programming standard developed by PGI, CRAY, CAPS, and NVIDIA for offloading programs written in C, C++, and Fortran from a host CPU to an attached accelerator device.

Similar to OpenMP, OpenACC requires you to annotate your source code with directives describing sections of your code and corresponding data that are to be executed on the GPU:

- For C, the directive is **`#pragma acc ...`**
- For Fortran, the directive is **`!$acc ...`**

To compile your code, you need to use a PGI compiler and options. For example:

```
module use /nasa/modulefiles/testing
module load comp-pgi/20.4

pgcc -acc -fast -Minfo your_program.c
```

See PGI Accelerator Compilers with OpenACC Directives for more information.

The Language-Based CUDA Model

The Compute Unified Device Architecture (CUDA) was created by NVIDIA and implemented on the NVIDIA GPUs. With CUDA, code developers write programs in C, C++, and Fortran, and incorporate CUDA extensions and optimized libraries. You can use either of the following programming environments:

- CUDA C is the original CUDA programming environment developed by NVIDIA for GPUs. It allows direct programming of the GPU from a high-level language. If you choose to write or rewrite portions of your code in CUDA C, you will need to load a **`cuda`** module and use the NVIDIA CUDA C compiler (**`nvcc`**) to build the executable. For example,

  ```
  module load cuda/10.2
  nvcc your_program.cu
  ```
- CUDA Fortran, a product of a collaboration between PGI and NVIDIA, includes a Fortran 2003 compiler and tool chain for programming NVIDIA GPUs using Fortran. To use CUDA Fortran, you will need to load a PGI compiler module and use the compiler **`pgfortran`** to build the executable. For example:

  ```
  module use /nasa/modulefiles/testing
  module load comp-pgi/20.4

  pgfortran your_program.cuf
  ```

  If the filename of your program does not end with the **`.cuf`** or **`.CUF`** suffix, use the **`-Mcuda`** option. For example:

```
pgfortran -Mcuda your_program.xx
```

See CUDA Fortran for more information.

TIP: There are many example codes for various disciplines available in the **/nasa/cuda/10.2/samples** directory on Pleiades. If you want to learn more about CUDA programming, the following resources may also be helpful: the online CUDA programming guide, and the book CUDA by Example: An Introduction to General-Purpose GPU Programming.

The Language-Based OpenCL Model

The Open Computing Language (OpenCL) is a framework for writing programs that run across heterogeneous platforms including CPU, GPU, and others. It consists of the OpenCL C language specification and OpenCL runtime API specification. Although OpenCL is C-based, there has been further development to bridge OpenCL and Fortran, such as FortranCL and CLFORTRAN.

Here is an example of building an executable where the source code for the host, written in C++, reads a code for the GPU that is written in OpenCL:

```
g++ -c -I /nasa/cuda/10.2/include host_program.cpp -o your_program.o
g++ host_program.o -o host_program.exe -L /nasa/cuda/10.2/lib64 -lOpenCL
```

If you have any questions or need assistance, contact the NAS Control Room at (800) 331-8737, (650) 604-4444, or support@nas.nasa.gov.

Requesting GPU Resources

GPU Node Reservation: Until April 30, 2022, thirty Cascade Lake-V100 GPU nodes (cas_gpu) are reserved for a special project. During this time, we recommend using the Skylake-V100 GPU nodes (sky_gpu) for your jobs requiring V100 GPUs.
There are currently two types of GPU nodes available: NVIDIA Kepler K40 GPU nodes, and NVIDIA Volta V100 GPU nodes. This article describes how to request each type for your PBS job.

Requesting K40 GPU Nodes (san_gpu)

In your PBS script or **qsub** command line, specify the **san_gpu** model type and **k40** queue name. Each **san_gpu** node is treated as one unit for exclusive access by a single job.

```
#PBS -l select=xx:ncpus=yy:model=san_gpu
#PBS -q k40
```

Requesting V100 GPU Nodes (sky_gpu and cas_gpu)

The method for requesting V100 GPU nodes changed on July 16, 2020. Instead of each node being treated as one unit for exclusive access by a single job, the nodes are now logically split into two vnodes, one for each socket and its associated CPU cores, GPUs, and memory. These resources are now sharable by different jobs. For more detailed information, see Changes to PBS Job Requests for V100 GPU Resources.

Important: To support these changes and new features, the V100 GPU nodes are currently being served by PBS server pbspl4, until further notice. You can either add **#PBS -q *queue_name*@pbspl4** in your PBS script or the **qsub** command line, or you can **ssh** directly from a Pleiades front end (PFE) to pbspl4 to submit a job.

Access to the V100 GPU nodes is provided through the **v100** queue and the **devel** queue. Each is described below.

WARNING: Your requested amount of CPU memory will be enforced. Once your job uses all of your requested amount of memory, it will be terminated.

v100 Queue

All of the **sky_gpu** and **cas_gpu** nodes that contain four V100 cards are accessed through the **v100** queue at all times.

The two **sky_gpu** nodes that contain eight V100 cards are accessed through the **v100** queue on weekends only; during the week, they are accessed through the devel queue.

To find default resources assigned to a job, run:

```
pfe% qstat -fQ v100@pbspl4 | grep default
```

The following examples will allocate resources in the **sky_gpu** nodes. Change **model=sky_gpu** to **model=cas_gpu** if you want to use the **cas_gpu** nodes instead. Additionally, the maximum number of CPUs you can request for Cascade Lake-based GPUs is 48 (**ncpus=48**) per node, whereas the max for Skylake-based GPUs is 36.

This request will result in 1 CPU, 1 GPU, and 32 GB of CPU memory:

```
#PBS -q v100@pbspl4
with
#PBS -lselect=1:model=sky_gpu
```

or

```
#PBS -lselect=1:model=sky_gpu:ncpus=1:ngpus=1:mem=32GB
```

The minimum number of CPUs that can be allocated to a job is 1. The minimum amount of memory is 256 MB.

To request exclusive access to *N* number of nodes, run:

```
#PBS -q v100@pbspl4
#PBS - lselect=N:model=sky_gpu:ncpus=36:ngpus=4:mem=360GB
#PBS -lplace=scatter:exclhost
```

Note: Constraints for the **`v100`** queue are currently set as:

- Two jobs per user in the running or queued state.
- Maximum of 16 nodes per job.
- Maximum of 24 hours walltime.

devel Queue

For development work between 8:00 a.m Monday and 5:00 p.m. Friday (Pacific Time), the two **`sky_gpu`** nodes that contain eight V100 cards are reserved for the **`devel`** queue, with a maximum of two hours and two nodes in a job.

The syntax for requesting resources via the **`devel`** queue is the same as that described for the **`v100`** queue, except the queue name is **`devel`** and the maximum value for **`ngpus`** is 8.

Reminder: When you run GPU jobs from the **`devel`** queue, you must use the PBS server pbspl4.

TIP: You can verify the CPUs and/or GPUs assigned to your job by using one of these options:

- ***GPU_hostname*% /u/scicon/tools/bin/clist.x -g**
- ***GPU_hostname*% nvidia-smi**

For detailed instructions describing how to request V100 GPU nodes, see Changes to PBS Job Requests for V100 GPU Resources.

Checking Job Status

To check the status of your jobs submitted to the **`k40`** queue, the **`v100`** queue, or the **`devel`** queue, run the **`qstat`** command on the specified host as follows:

```
pfe% qstat k40 -u your_username

pbspl4% qstat v100 -u your_username
or
pfe% qstat v100@pbspl4 -u your_username

pbspl4% qstat devel -u your_username
or
pfe% qstat devel@pbspl4 -u your_username
```

Filesystems

Pleiades Home Filesystems

The home filesystems on Pleiades are HPE/SGI NEXIS 9000 filesystems that are NFS-mounted on all of the Pleiades front-end (PFE) nodes and compute nodes. Each NAS user is provided a home directory on the Pleiades home filesystems, which are the home filesystems for Pleiades, Aitken, Electra, and Endeavour. After you are granted an account, your home directory is set up automatically during your first login.

Your home directory has a quota of 8-10 GB of storage. For temporary storage of larger files, use the Lustre /nobackup filesystems. For long-term storage, use the Lou mass storage system.

Quota Limits and Policy

Disk space quota limits are enforced on the Pleiades home filesystems. By default, the soft limit is 8 GB and the hard limit is 10 GB. There are no inode limits on the home filesystem.

To check your quota and usage on your home filesystem, run the **`quota -v`** command as follows:

```
%quota -v
Disk quotas for user username (uid xxxx):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
saturn-ib1-0:/mnt/home2
                7380152  8000000 10000000          190950       0       0
```

The NAS quota policy states that if you exceed the soft quota (the number listed under **`quota`** in the above sample output), an email will be sent to inform you of your current usage and how much of the grace period remains. It is expected that you will occasionally exceed your soft limit; however, after 14 days, any attempts to write to the filesystem will result in error. Any attempt to exceed the hard limit (the number listed under **`limit`** in the above sample output) will result in error.

If you believe that you have a long-term need for higher quota limits on the Pleiades home filesystem, send an email justification to support@nas.nasa.gov. Your request will be reviewed by the HECC Deputy Project Manager for approval.

Note: For temporary storage of larger files, or a large number of files, use your Lustre /nobackupp*X* directory. For normal long-term file storage, transfer your files to the Lou mass storage systems.

See also: Quota Policy on Disk Space and Files.

TIP: If you receive the following error message when logging in, you won't be able to run X applications. This error is usually due to exceeding your home filesystem quota. To avoid the error, decrease your disk usage.

`/usr/X11R6/bin/xauth: error in locking authority file /u/`*`username`*`/.Xauthority`

Backup Schedule

Files on the home filesystem are backed up daily.

Pleiades Lustre Filesystems

Pleiades, Aitken, Electra, and Endeavour share several filesystems intended to provide working space for compute jobs. These filesystems, called "nobackup" (/nobackupp*X*), provide over 90 petabytes (PB) of disk space and serve many thousands of cores. The /nobackup filesystems are Lustre-based filesystems managed under Lustre software version 2.*x*.

Using the Lustre /nobackup Filesystems

As the name suggests, these "nobackup" filesystems are for temporary use, and are not backed up. Lustre can handle many large files, but you cannot use the Lustre filesystems for long-term storage. If you want to save your files, move them to Lou.

Each Lustre filesystem is shared among many users. To learn how to achieve good I/O performance for your applications and avoid impeding the I/O operations of other users, read the related articles listed at the bottom of the page.

Lustre filesystem configurations are summarized at the end of this article.

WARNING: Any files that are removed from Lustre /nobackup filesystems *cannot* be restored. You should store essential data on the Lou mass storage system (lfe[*5-8*]) or on other, more permanent storage.

Which Lustre Filesystem is Assigned to Me?

Once you are granted a NAS account, one of the Lustre filesystems will be assigned to you. To find out which one, run the **`ls`** command as follows:

```
pfe21% ls -l /nobackup/your_username
```

In the output, look for the filesystem name (**nobackupp*X***). For example, the following output shows that the user's assigned filesystem is /nobackupp17:

```
lrwxrwxrwx 1 root root 19 Sep 23  2021 /nobackup/your_username -> /nobackupp17/your_username
```

Default Quota and Policy on /nobackup Filesystems

Disk space and inode quotas are enforced on the /nobackup filesystems. The default soft and hard quota limits for inodes are 500,000 and 600,000, respectively. Quotas for the disk space are 1 terabyte and 2 terabytes, respectively. To check your disk space, inode usage, and quota on your /nobackup filesystem, run the **`lfs`** command as follows:

```
% lfs quota -h -u username /nobackup/username
Disk quotas for user username (uid nnnn):
        Filesystem      used       quota   limit   grace   files    quota   limit   grace
/nobackup/username        4k      1.024T  2.048T       -       1   500000  600000       -
```

It is expected that your data will occasionally exceed the soft limit. However, after a 14-day grace period, your access to the batch queues will be restricted and you will not be able to run new jobs until your data is reduced below the soft limit. If you reach the hard limit, your batch access will immediately be restricted, but any running jobs will continue.

If you anticipate a long-term need for higher quota limits, please send a justification via email to support@nas.nasa.gov. Your request will be reviewed by the HECC Deputy Project Manager for approval.

For more information, see Quota Policy on Disk Space and Files.

Note: When a Lustre filesystem is full, the jobs writing to it will hang. A Lustre error with code `-28` in the system log file indicates that the filesystem is full. NAS support staff will typically send emails to those using the most space, with a request to clean up their files.

Lustre Filesystem Configurations

The way your Lustre filesystem is configured determines how your files are striped. There are two types of configurations: Progressive File Layout (PFL), and the standard Lustre configuration. Most of the /nobackup filesystems have PFL configurations.

Progressive File Layout Configurations

Lustre's Progressive File Layout (PFL) feature enables filesystems to dynamically change the stripe count for files based on their size. This means that the files are dynamically striped, therefore, you should ***not*** set custom stripe size or stripe counts.

The Lustre PFL filesystems are /nobackupp[*10-13*] and /nobackupp[*15-19*]; among them, /nobackupp[17-19] are equipped with a small number of solid state drives (SSDs) in addition to the hard disk drives (HDDs), while /nobackupp[10-13,15-16] only have HDDs.

The HDD configurations of /nobackupp[10-13,15-19] are listed in the table below with the abbreviation nbp*X.* P = petabytes; T = terabytes; M = megabytes.

Lustre PFL HDD Configurations

Filesystem	nbp10	nbp11	nbp12	nbp13	nbp15	nbp16	nbp17	nbp18	nbp19
# of OSTs	46	69	23	23	23	23	32	32	16
size/OST	70.7 T	70.7 T	70.7 T	70.7 T	70.7 T	70.7 T	565 T	565 T	565 T
Total Space	3.2 P	4.8 P	1.6 P	1.6 P	1.6 P	1.6 P	18.6 P	18.6 P	9.3 P

For /nobackupp[10-13,15-16], which have only HDDs, a common default PFL stripe setting is used as follows:

Lustre PFL Stripe Counts per File Size

File Size	0 - 10 MB	10 MB - 17 GB	17 - 68 GB	> 68 GB
Stripe Count	1	4	8	16

For /nobackupp[17-19], different PFL configurations are set among the SSD and HDD pools. To learn more, see Progressive File Layout with SSD and HDD Pools.

Standard Configurations

The standard Lustre filesystems are /nobackupp1 and 2; in the table below, these are abbreviated as nbp*X.* P = petabytes; T = terabytes; M = megabytes.

Standard Lustre Configurations

Filesystem	nbp1	nbp2
# of OSTs	144	342

size/OST	43.6/57.2 T	43.6/71.2 T
Total Space	7.1 P	18 P
Default Stripe Size	1 M	1 M
Default Stripe Count	4	4